Slice Orchestration for Multi-Service Disaggregated Ultra Dense RANs

Chia-Yu Chang, Navid Nikaein, Osama Arouk, Kostas Katsalis, Adlen Ksentini, Thierry Turletti, Konstantinos Samdanis

Slice Orchestration for Multi-Service Disaggregated Ultra Dense RANs

Chia-Yu Chang*, Navid Nikaein*, Osama Arouk†, Kostas Katsalis‡,
Adlen Ksentini*, Thierry Turletti†, Konstantinos Samdanis‡
*EURECOM, France, †Inria, Université Côte d'Azur, France,
‡Huawei Technologies, Co.,Ltd, Germany

Abstract

Ultra Dense Networks (UDNs) are a natural deployment evolution for handling the tremendous traffic increase related to the emerging 5G services, especially in urban environments. However, the associated infrastructure cost may become prohibitive. The evolving paradigm of network slicing can tackle such a challenge while optimizing the network resource usage, enabling multi-tenancy and facilitating resource sharing and efficient service-oriented communications. Indeed, network slicing in UDN deployments can offer the desired degree of customization in both vanilla Radio Access Network (RAN) designs, but also in the case of disaggregated multi-service RANs. In this article, we devise a novel multi-service RAN environment, i.e., RAN runtime, capable to support slice orchestration procedures and to enable flexible customization of slices as per tenant needs. Each network slice can exploit a number of services, which can either be dedicated or shared between multiple slices over a common RAN. The novel architecture we present concentrates on the orchestration and management systems. It interacts with the RAN modules, through the RAN runtime, via a number of new interfaces enabling a customized dedicated orchestration logic for each slice. We present results for a disaggregated UDN deployment where the RAN runtime is used to support slice-based multi-service chain creation and chain placement, with an auto-scaling mechanism to increase the performance.

Index Terms

Network Slicing, RAN slicing, 5G, Orchestration, Service Chaining, Network Embedding.

I. Introduction

To cope with emerging services and the associated data traffic volume increases, a key trend toward Fifth Generation (5G) deployments is network infrastructure densification, known as Ultra-Dense Networking

(UDN) [1]. In UDN, base stations (BSs) can be located on every lamp post, bus stop, or indoor environments, providing seamless coverage, while coping with traffic overloading. However, the deployment and operational costs may be too high, since UDNs were originally conceived in the context of small cell networks, considering legacy 4G BSs. A step towards reducing such costs came with the design of Cloud-RAN (C-RAN) [2], where the monolithic BSs were decomposed into (i) distributed radio elements with much smaller footprints and (ii) remote pools of baseband units that centralize the remaining RAN functions. Such initial C-RAN concept now evolves towards a more flexible deployment by disaggregating RAN entities into Radio Unit (RU) equipment, Distributed Unit (DU) and Centralized Unit (CU) as discussed in the Third Generation Partnership Project (3GPP). The concept of disaggregated RAN retains the benefits of *centralization* to enable a coordinated and cooperative processing. Additionally, it allows a flexible deployment of services at the DU and CU located in cloud infrastructures offering a simpler network *densification* at the RU level.

To reduce costs even further, UDN deployments may support multi-tenancy enabling infrastructure sharing, while optimizing resource utilization by offering isolation and network customization on per-tenant basis. This can be realized with the adoption of the emerging network slicing paradigm. Network slicing enables the creation of self-contained logical networks on top of shared physical and virtualized infrastructures. The resources allocated for a slice can be completely isolated, e.g., a different set of spectrum and cell sites, or partially isolated with certain resources being shared such as the radio spectrum. According to 3GPP TR28.801 a Network Slice Instance (NSI) includes a set of network functions and the resources for these network functions spanning multiple subsystems, i.e., Core Network (CN) and RAN, which are arranged and configured, forming a complete logical network that meets certain network characteristics required by a service instance. For each NSI, both user-plane (UP) and control-plane (CP) functions can be customized to satisfy specific service demands. *Softwarization*, *virtualization*, and *disaggregation* are key enablers to flexibly customize a slice and satisfy the requirements of end-to-end (E2E) services when deploying UDNs.

For the mobile network, a crucial aspect of E2E network slicing is related to the RAN-domain. RAN slicing solution prototypes have been developed, exploiting cloud computing, Software-Defined Networking (SDN) and Network Function Virtualization (NFV) technologies [3], [4]. Nevertheless, in the case of disaggregated multi-service RAN for UDN deployments, the field of network slice orchestration and the interactions between the orchestrator with the underlay RAN is not explored yet. When orchestrating multiple services at the RAN, efficient resource usage and service scaling are mandatory, while considering specific slice requirements in terms of performance and customization.

To this end, we identify the following three challenges to design slice-enabled disaggregated RANs:

1) Efficient service modeling to validate service requirements when deploying on per-slice basis;
2) Multi-service chaining of customized and/or shared physical/virtual network functions (PNFs/VNFs);
3) Multi-service chain placement algorithm in densely-deployed infrastructures with auto-scaling action.

To tackle these challenges, we propose a novel multi-service system, denoted as RAN runtime, which can be exploited by the orchestrator. The RAN runtime supports all the necessary functionalities on top of RAN modules, for both monolithic and disaggregated RANs facilitating shared or dedicated RAN services. A set of interfaces are defined for the interaction between the RAN runtime and the orchestration system. Our proposal retains the compatibility with the E2E service orchestration model introduced in 3GPP TR28.801, and further extends it to support multi-service operations for the disaggregated RAN. Moreover, we provide the dynamic service modeling approach and propose a multi-service chaining and placement algorithm for the disaggregated RAN. Finally, we evaluate the proposed algorithm in a realistic UDN scenario showing the impact of the auto-scaling actions to increase the service acceptance ratio.

II. Related work

Network densification is one enabling technology to accommodate high traffic volume since the early deployments of 4G with the introduction of small cell networks [1]. A natural UDN evolution is towards C-RAN providing a key option leveraging operational benefits in terms of resource management and interference control, while reducing the infrastructure costs by centralizing RAN functions into cloud platforms [2]. Sharing the RAN and cloud resources, and optimizing operations with respect to particular services can improve the efficiency and costs of UDNs. The emerging network slicing concept can enhance service optimization and support multi-tenancy via the means of customization and isolation [5]. 3GPP addressed network slicing from the architecture perspective in TS23.501, studied the related management and orchestration in TR28.801, and highlighted the RAN slicing aspect in TR38.801. An E2E network slicing overview with respect to different network domains is introduced in [6]. Also, a joint RAN and transport network slicing approach facilitating the programmable control/orchestration plane is provided in [7].

The process of RAN slicing should fulfill 5G RAN design requirements [8] through a flexible CP and UP decoupling that relies on RAN programmability. The notion of RAN programmability is explored by xRAN[1] aiming to bring the software-based, extensible RAN to light, while FLEXRAN[2] platform focuses on the software-defined RAN (SD-RAN) concept and utilizes northbound application programming protocol (API) to enable RAN monitoring and control. In line with the SD-RAN paradigm, several

[1] http://www.xran.org/ [accessed on 13-Mar-2018] [2] http://networks.inf.ed.ac.uk/flexran/ [accessed on 13-Mar-2018]

RAN slicing studies are initiated. A RAN slicing architecture in [3] enforces radio resource management relying on a resource visor per slice. The BS hypervisor in [4] simultaneously isolates slice-specific control logic and shares radio resources, while in [9], a RAN controller guides the inter-slice resource allocation based on a predetermined policy. The network slice brokering solution in [10] provides traffic forecasting, admission control and scheduling.

Regarding network orchestration, the European Telecommunications Standards Institute (ETSI) Management and Orchestration (MANO) architectural framework may collect functional blocks, data repositories and interfaces for orchestrating virtualized infrastructures and VNFs. Several open-source implementations targeting mobile networks can be found such as OSM[3], OPNFV[4], M-CORD[5], ECOMP[6], and JoX[7]. These solutions allow slice customization; however, they only consider isolation by running the RAN service in a virtualized environment and lack to meet the trade-off challenge between customization and sharing. By contrast, the proposed RAN runtime provides an environment to incorporate flexibility to satisfy different requirements and orchestrate services for densely-deployed disaggregated RANs.

III. RAN RUNTIME SLICING SYSTEM

We hereby outline the slice-based orchestration architecture, detail the proposed RAN runtime system and elaborate on the functionalities of the required interfaces for the communication between the RAN runtime and the orchestration systems.

A. Architecture Overview

Network slicing provides an E2E connectivity analyzing the service requirements to ensure the desired performance on a continuous basis. Such process consists of the service management operation that handles procedures related to admission control, charging and service creation. Besides, a network slice management operation handles the instantiation and life-cycle management of a network slice considering also CP processes such as slice selection and multi-slice connectivity. 3GPP defines the following entities in TR28.801 regarding orchestration and management of NSIs:

- *Communication Service Management Function (CSMF)*: Responsible for translating the communication service requirements to network slice requirements.
- *Network Slice Management Function (NSMF)*: Responsible for the E2E management and orchestration of the NSI.

[3] https://osm.etsi.org/ [accessed on 13-Mar-2018] [4] https://www.opnfv.org/ [accessed on 13-Mar-2018] [5] https://www.opennetworking.org/solutions/m-cord [accessed on 13-Mar-2018] [6] http://about.att.com/innovation/labs [accessed on 13-Mar-2018] [7] https://gitlab.eurecom.fr/mosaic5g [accessed on 13-Mar-2018]

- *Network Slice Subnet Management Function (NSSMF)*: Responsible for the management and orchestration of the sub-network slice instance in a specific domain, e.g., RAN-NSSMF and Core-NSSMF is responsible for the management of RAN and CN respectively.

Fig. 1 illustrates the proposed orchestration architecture, which includes the NSSMF entity to carry out the RAN life-cycle management and the RAN runtime to provide a multi-service execution environment considering both dedicated and shared orchestration functions for a monolithic or disaggregated node. NSMF is responsible for preparing an integrated environment for the NSI, while also controlling the NSI life-cycle, interacting with the domain-specific orchestrator, i.e., RAN-NSSMF. The latter contains a slice-specific subsystem control and management. For UDN deployments, the RAN runtime is introduced to provide a unified and flexible execution environment used to run multiple virtualized RAN instances with the required level of isolation and sharing, operating over the underlying RAN modules and resources.

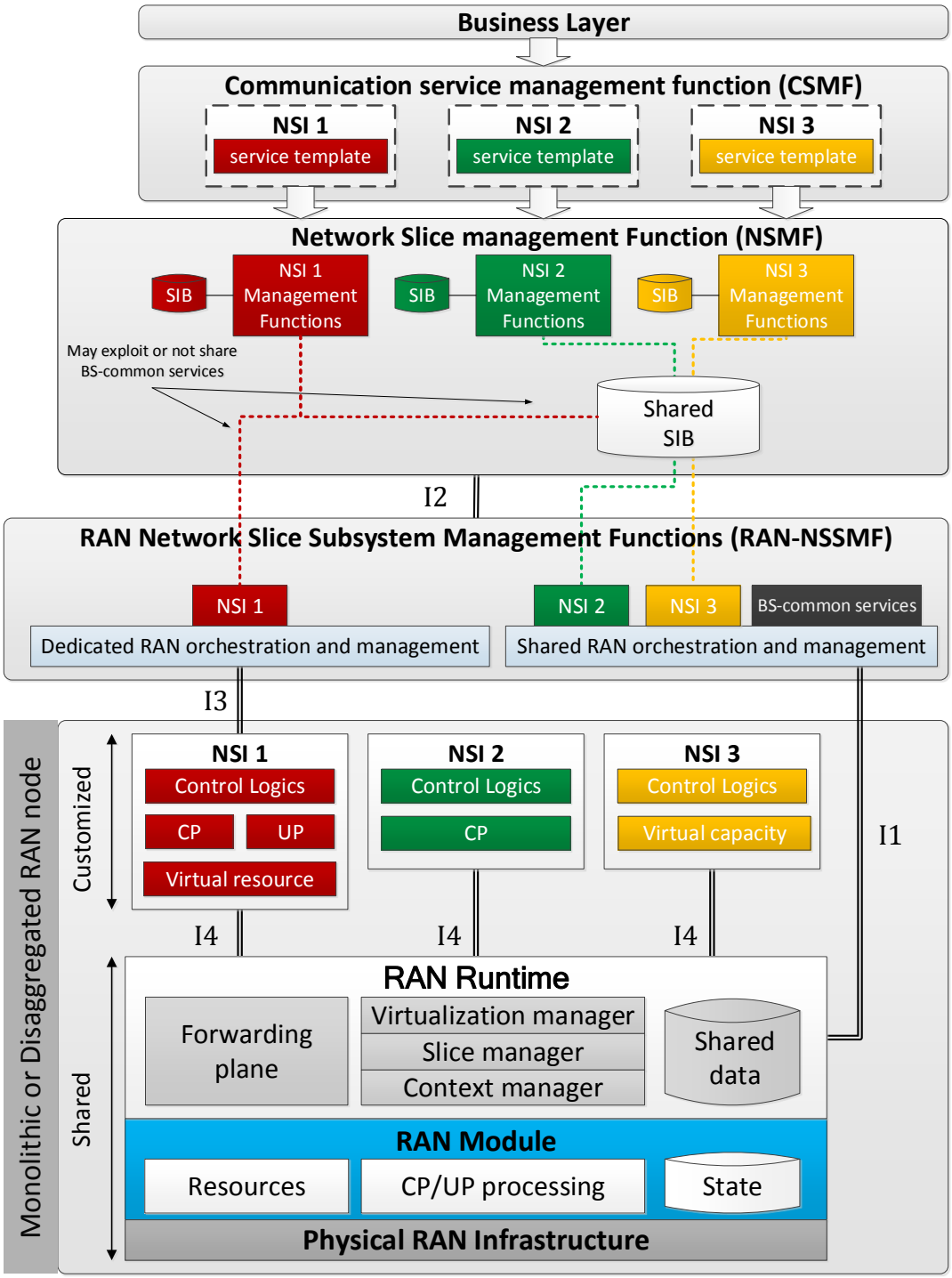
B. Design elements of the RAN runtime slicing system

The proposed RAN runtime element is shown in Fig. 1. It allows a running NSI to interact with the RAN modules via the NSMF, which monitors the allocated resources and controls the behavior of the underlying network.

The RAN module is a subset of RAN functions together with the associated resources and their states, which performs a portion or all of RAN operations over the common and/or specialized RAN infrastructure. The RAN runtime allows 3rd party tenants to (a) control their NSIs via the means of NSMF, (b) introduce a customized control logic (e.g., handover decision) or customized CP/UP processing, (c) operate on a set of virtual resources (e.g., resource blocks or frequency spectrum) or capacity (e.g., data rate), and (d) access their CP/UP state. It also enables the operator to manage the underlying RAN module, enforce slice-specific policies, perform access control, and expose BS services.

The isolation, abstraction and customization properties provided by the RAN runtime enable the RAN-NSSMF to orchestrate the RAN NSI and the behavior of the underlying RAN modules considering the service requirements. The RAN runtime also allows the creation of NSIs that exploit reusable RAN service chains reducing instantiation costs and CP/UP processing. Within the RAN runtime, the following two services are provided to facilitate the NSI orchestration procedures:

- The *Slice manager* determines the CP/UP processing chain for each NSI and the relevant traffic flows based on the service requirements and its context. It also programs the forwarding plane through a set of rules allowing to direct the input and output streams across shared processing functions operated by the underlying RAN module, while customizing the processing function required by each NSI. When NSI is modified, the RAN functions operated by the *slice manager* may be updated



Interface	Description
I1	Expose active RAN runtime services and retrieve messages for monitoring and feedback.
I2	Subscribe for the slice-specific events and populate SIB accordingly.
I3	Customize management and monitoring for each slice and indicate any changes in underlay RAN.
I4	Register a slice to the RAN runtime and consume RAN runtime service as a separate process.

Fig. 1: Architecture of RAN runtime slicing system and E2E service orchestration.

to support service continuity. Such manager also resolves conflicts among different NSIs based on predetermined policy rules.

- The *virtualization manager* provides the required level of isolation and sharing to each NSI. Specifically, it abstracts physical resources to virtualized ones and partitions resources based on NSI requirements, modules context and state.

Our approach relies on the two-level scheduler framework introduced in [3]. The resources are abstracted adopting the concept of virtual resource blocks (vRBs), i.e., independent from physical resource blocks (PRBs), advertising the preserved size information. The vRB pool allows each slice to customize its scheduler function. With the means of vRB, the exact placement of the physical resources are abstracted from the slice scheduler, effectively providing resource isolation among different tenants. Such scheduler relies on the information provided by the *virtualization manager*, such as channel state information. Once the vRB are allocated to a user, the virtualization manager maps the vRB/users allocation to PRB considering the priority of each slice and service level agreement (SLA). Thus, it ensures that the PRB resources are shared respecting the corresponding policy.

Both the *slice manager* and the *virtualization manager* exploit the data that shared between different NSIs. This data is related to the different NSIs context (e.g., basic information like its identity, registered RAN runtime services, user context and their NSI association) and module context (e.g., CP and UP state information, module primitives) that is used to customize and manage a slice. Such data may be exploited by different NSIs especially once modified to reflect user and network dynamics, i.e., changing the RAN functional split between CU, DU and RU [11].

Moreover, the RAN runtime can support a number of slice services depending on the amount of requested resources and SLA of each slice. If all slices opt for a virtualized resource without any hard guarantee, then a large number of slices can be accommodated subject to the overhead at RAN runtime to manage slices. Otherwise, the bottleneck will be the number of requested radio resources and traffic dynamics when a certain level of guarantee is required by the running slice.

The NSMF maintains a shared slice information base (SIB) that contains slice specific information including SLA, user context, user to slice association and service inventory. The CSMF creates a service template for each slice that communicates to NSMF, which contains tangible slice requirements. Using such template, the NSMF performs the NSI preparation. Once instantiated, each service is orchestrated in RAN-domain, which can be in either shared or dedicated mode from the RAN-NSSMF entity. For NSI 1 in Fig. 1, the RAN-NSSMF orchestrates a set of dedicated RAN functions that are customized according to its SLA requirements. The RAN runtime is the intermediate entity responsible to fine tune the necessary RAN modules. As NSI 2 and NSI 3 do not request such customization, their functions are

managed through a shared service from the RAN runtime, exploiting also the information stored in a shared SIB at the NSMF level.

C. Interfaces functionality analysis

A set of interfaces (I1 to I4) are introduced enabling communication between the various entities as depicted in Fig.1.

I1 serves several purposes. Before instantiating any NSI or service, the RAN runtime and RAN modules are operational and expose BS services like broadcasting system information or neighboring nodes information. These are irrelevant to the number of instantiated NSIs. Information about the active services and capabilities of the RAN runtime are exposed through the I1 interface. A service registration to the *slice manager* of the RAN runtime during the creation of a NSI is required. Based on the service templates, the *slice manager* performs several operations as detailed in section III-B. However, certain operations are only performed for slices orchestrated and managed in a shared manner. Monitoring and feedback messages are retrieved from the RAN runtime through I1 to facilitate coordination among different slices regarding the available resources and service requirements.

I2 is the interface between the NSMF and the RAN-NSSMF and is currently standardized by 3GPP.

I3 is the interface between the dedicated orchestrator and the corresponding dedicated NSI functions at the RAN node, through which a slice owner can customize service management and monitoring. For instance, it can be used for the communications required to operate customized CP/UP processing and then program the forwarding plane at the RAN runtime accordingly (through I4 interface).

I4 provides a communication channel with the RAN runtime allowing a NSI to register to the RAN runtime and consume its services, whether it is local or remote. When a slice is deployed locally, I4 exploits the inter-process communication mechanism to allow a slice to perform real-time operations (e.g., Medium Access Control (MAC) function) with hard guarantees. However, when considering non-time-critical operations (e.g., Packet Data Convergence Protocol (PDCP) function), communication is made through an asynchronous communication interface (e.g., message bus).

IV. Slice orchestration for disaggregated ultra-dense RAN

While the RAN runtime enables a multi-service architecture, slice orchestration in a multi-tenant RAN remains an open question. In this section, three main challenges are investigated: (1) *dynamic service modeling* to optimize the RAN service template, (2) *multi-service chaining* to customize per-slice service chain, and (3) *multi-service placement* for shared/dedicated service chains.

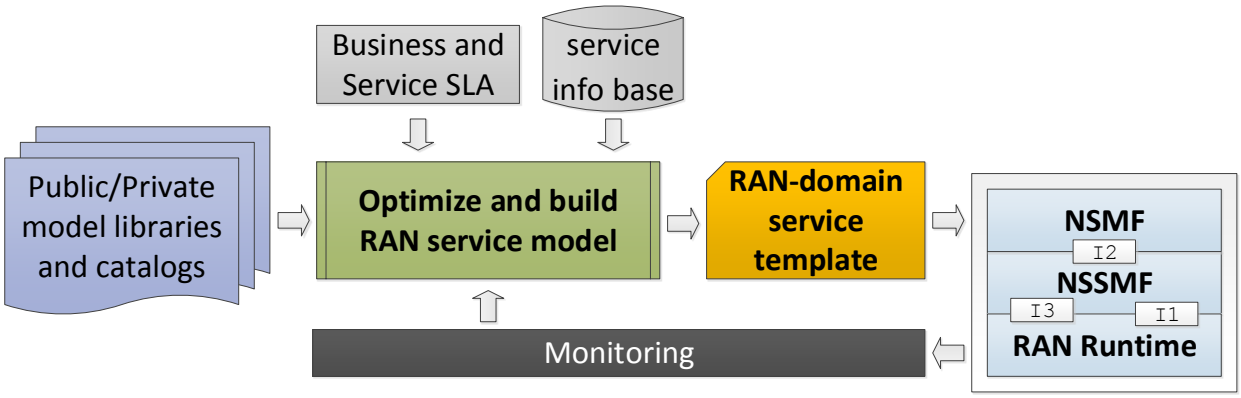


Fig. 2: Process of RAN modeling and service template optimization.

A. *Service modeling*

While service templates capture use-case specific requirements with the associated SLA and operator policies [12], dynamic service updates may be needed to maintain and optimize the performance. As depicted in Fig. 2, a slice template contains different types of resources (e.g. clouds, nodes, links, network functions, radio spectrum, and RAN modules) considering operators service catalogs and requested SLA. During the runtime phase, the service template may be periodically optimized (e.g., compare service key performance indicators (KPIs) against the SLA) based on the monitoring information collected through `I1` and `I3` interfaces provided by the RAN runtime. The updated service template is then pushed by the orchestrator, actuated by NSMF, and applied to the RAN infrastructure through RAN-NSSMF. Such optimizations may require negotiations among different providers to fulfill the service requirements [12], and can be applied for resource partitioning (by the RAN runtime through the orchestrator), service placement and scaling (by orchestrator), functional split (by operator), processing customization (by slice owner) relying on predefined rules and/or cognitive logic.

B. *Multi-service chaining and placement*

In the following, we detail how the RAN functions are chained and placed across shared and customized CP/UP processing on a per-slice basis.

1) Multi-service chaining: A RAN service chain can be functionally split (e.g., option 1 to 8 described in 3GPP TR38.801) in disaggregated deployments. Such service chain can be composed *horizontally* based on the shared functions provided by the underlying RAN module (See Fig. 3) and/or *vertically* through the customized CP/UP processing delivered by the slice owner to fulfill its service requirements. Note that the functional split between disaggregated RAN entities is generally determined by the operator based on

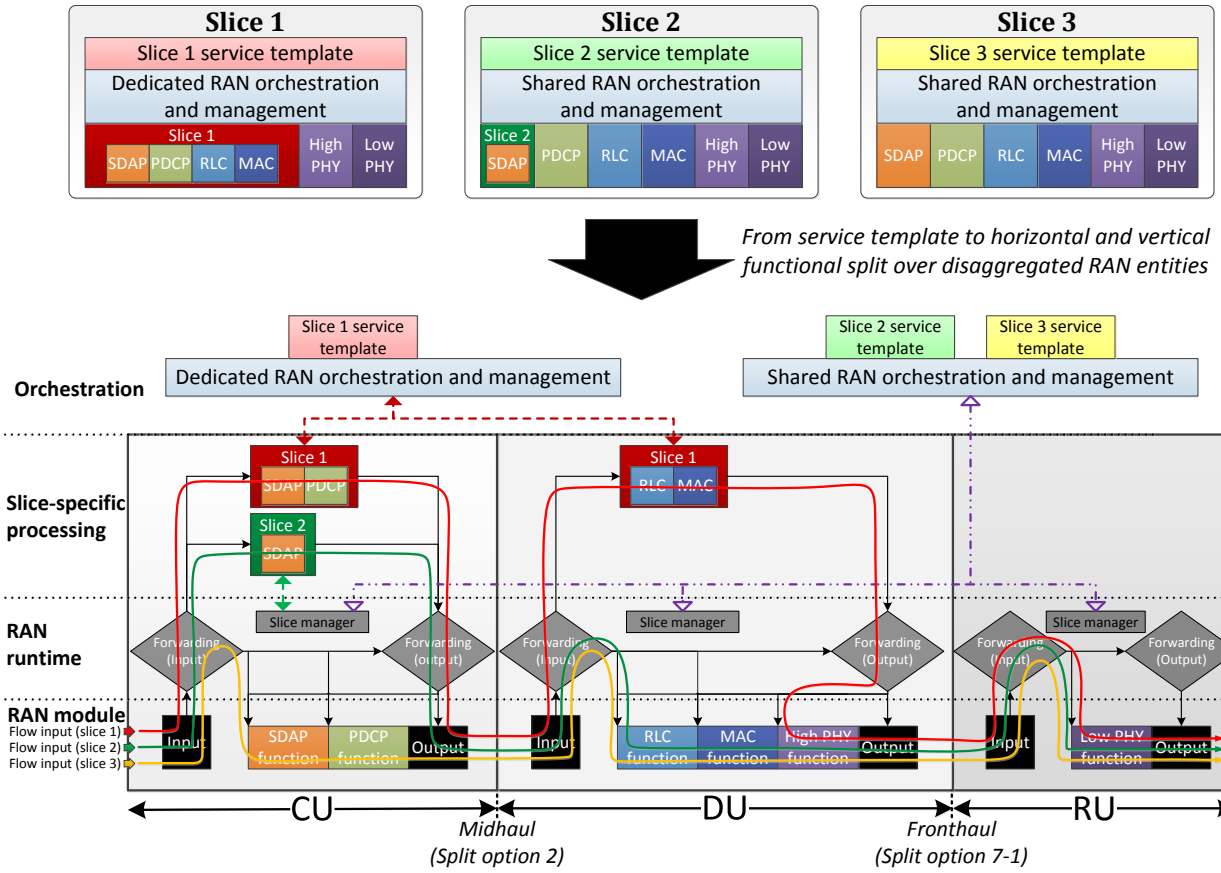


Fig. 3: Example of multi-service chaining and forwarding in a disaggregated RAN.

the aggregated BS KPIs and fronthaul/midhaul performance, e.g., capacity and latency. However, slice owners can change the functional split through the CP/UP customization with the associated input/output forwarding chain. For instance, a service may request a hardware-based accelerator for channel decoding and a dedicated radio resource scheduling to enable low-latency communications. Note the network functions are managed by either the RAN runtime or the VNF manager, corresponding to PNF and VNF respectively. The E2E RAN service chain is maintained by the forwarding plane at the RAN runtime, leveraging the SDN-based match-action abstractions to build slice-specific forwarding input and output paths across shared and dedicated functions. A disaggregated RAN example with different levels of slice customization in downstream path is shown in Fig. 3. The first slice requires a dedicated service management and orchestration with customized functions over Service Data Adaptation Protocol (SDAP), PDCP, Radio Link Control (RLC), MAC, and High Physical (High-PHY) layers while shared functions at the Low-PHY layer. The second slice utilizes the shared service orchestration and customizes the SDAP layer functions. Slice 3 is built on the top of the shared network function chains over all disaggregated

RAN entities.

Note that the slice-specific function customization is described in the aforementioned service template, and the customized forwarding path of each slice is managed by the *slice manager* of RAN runtime under the control of the corresponding RAN-NSSMF. However, the input/output endpoints of each disaggregated RAN entity perform infrastructure-dependent packet processing like encapsulation and switching/routing for fronthaul/midhaul transportation. As the RAN runtime maintains the state for both dedicated and shared network functions, it facilitates dynamic service template update by handling state synchronization among the involved RAN entities. This is the case when the RAN functional split and/or placement are updated following the spatio-temporal traffic dynamics.

Moreover, different failures may happen at several levels when chaining a RAN service, e.g., broken infrastructure, lack of resources, missing SLAs, or RAN runtime overload. The failure can also be service-dependent: if it is customized, then the simplest recovery approach would be reusing the shared chain with the default configurations. The consequence would be service unavailability for corresponding users, but they may still remain connected to the network for basic BS services.

2) Multi-service placement: Once RAN service chains are composed, the associated network functions are placed accordingly while respecting the service requirements. Such requirements are described in terms of resources (e.g., compute, memory, networking) and performance (e.g., average throughput, maximum latency). The placement also considers objectives such as cost/power/resource optimization imposed by the operator.

We propose a two-stage placement algorithm as the slice service chains are composed both horizontally and vertically. Such algorithm extends the Multi-Objective Placement (MOP) one described in [13] and includes the following steps:

- **Step 1**: For each shared function with distinct latency constraint in the chain, determine its eligible regions (ERs) corresponding to the set of RAN nodes that satisfy the latency requirements;
- **Step 2:** Determine the candidate group (CG) comprising the nodes from ERs satisfying the remaining slice requirements;
- **Step 3:** Select the best node (BN) among CG based on the considered operator objective, e.g., load balancing;
- **Step 4:** Repeat the above three steps to place the customized functions based on the results of the shared chains.

An example is shown in Fig. 4 with 14 nodes (i.e., $\{N_1, \cdots, N_{14}\}$) when placing the function chain of CU ($\{\text{SDAP}, \text{PDCP}\}$) and DU ($\{\text{RLC}, \text{MAC}, \text{High-PHY}\}$), which are categorized into 3 levels: $\{N_1, N_2\}$, $\{N_3, \cdots, N_6\}$ and $\{N_7, \cdots, N_{14}\}$. All densely-deployed M RUs are grouped into several RU groups,

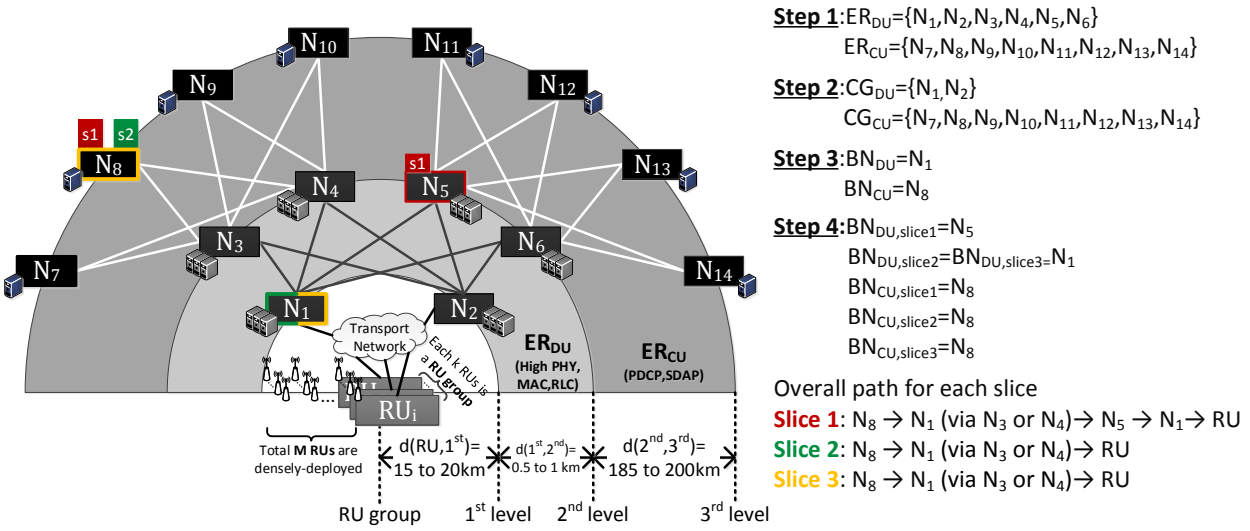


Fig. 4: Example of a two-stage function placement for multi-service.

each with k RUs that is associated with a pair of (CU,DU). The algorithm firstly selects the ERs based on the latency constraints of service function chains to be placed at CU and DU (cf. Step 1 in Fig. 4). Then, CGs are formed based on the total processing requirement of a given function chain (cf. Step 2 in Fig. 4), and BNs are selected to place the shared functions chain (cf. Step 3 in Fig. 4). Afterwards, the customized processing for each slice is placed based on the results of shared chain following the same algorithm (cf. Step 4 in Fig. 4). As the input and output endpoints are not slice-customized, an extra round-trip-time (RTT) is included when computing the ER of customized processing to capture the infrastructure-dependent packet processing. Taking the DU in Fig. 4 as an example, the placement of customized functions of slice 1 (e.g., RLC, MAC) shall preserve the latency constraint with respect to the remaining shared chain (e.g., Input, High-PHY, Output). Thus, the RTT between N_1 and N_5 is considered when placing the customized chain at N_5 (cf. the overall path in Fig. 4).

V. Performance Evaluation

We analyze the performance of the proposed multi-service chaining and placement approach considering a UDN deployment based on the processing time data obtained from the OpenAirInterface[8] platform. The auto-scaling actions is highlighted in response to the network dynamics.

A. Experiment scenario

We consider the three-level infrastructure topology shown in Fig. 4 with the following inter-level distances: d(RU,1st)=15km, d(1st,2nd)=0.5km, d(2nd,3rd)=185km. A subset of M RUs are grouped together,

[8] https://www.openairinterface.org [accessed on 13-Mar-2018]

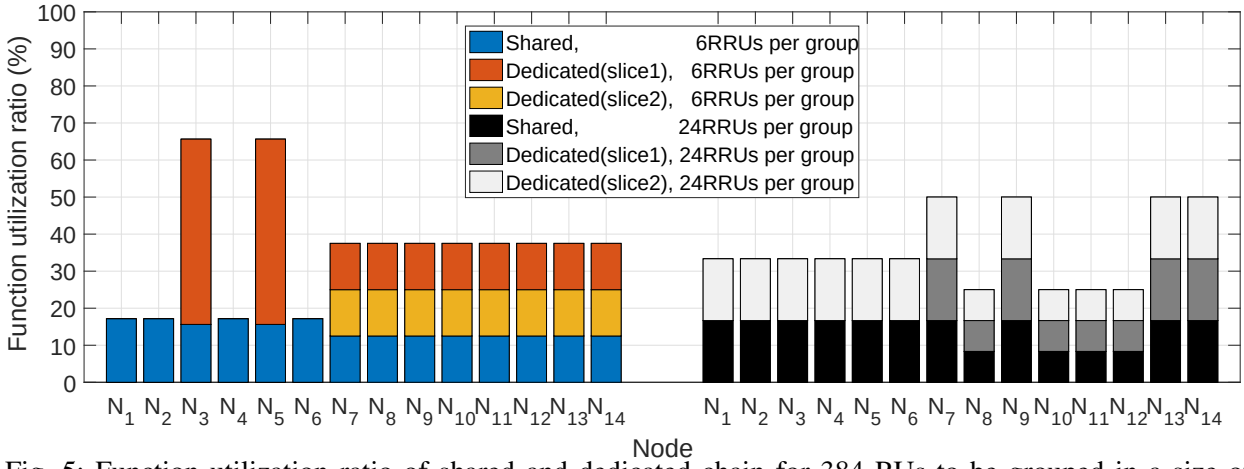


Fig. 5: Function utilization ratio of shared and dedicated chain for 384 RUs to be grouped in a size of 6 (left) or 24 (right) RUs.

where each group forms the minimum placement granularity for a given chain as depicted in Fig. 4. For instance, if 6 RUs are grouped together, then 6 function chains are placed simultaneously across DU and CU such that they are physically co-located within the same node. These chains within a group can facilitate real-time control and coordination like joint processing to enable the Coordinated Multi-Point (CoMP) feature, which is important in a UDN to improve the network performance. Moreover, we apply the same service chain (i.e., functional split, customized functions) for each slice as in Fig. 3.

B. Simulation Results

Fig. 5 shows the function utilization ratio of all 14 nodes in the scenario with 384 RUs that are grouped in two different sizes: i) 6 RUs, forming 64 RU groups and ii) 24 RUs, forming 16 RU groups. Hence, there are 384 densely-deployed chains to be embedded over these 14 nodes with different numbers of CPU: N_1 to N_6 each with 32 CPUs, and N_7 to N_{14} each with 2 CPUs. We observe that the shared chains are evenly distributed among the 14 nodes, with slightly better allocation when grouping 6 RUs as it has a lower level of granularity. The dedicated chains at CUs (i.e., N_7 to N_{14}) are also uniformly distributed for both RU groups and are collocated with the shared chain for slice 1 and slice 2. The reason being that the input/output endpoints are infrastructure-dependent and the RTT between any two nodes at level 3 breaks the latency constraint of the chain. Such issue can be solved by either enabling slice-customized input/output endpoints (e.g., slice 1 may move dedicated chain and input/output to N_9), or provisioning additional links between nodes of level 3 (e.g., between N_8 and N_7). In contrast, the results of DU (i.e., N_1 to N_6) show different trends for two RU group sizes. When grouping 24 RUs, the requested DU resources restrict the possible placement locations and cause a higher probability to collocate the dedicated function with the shared one, unlike when grouping 6 RUs.

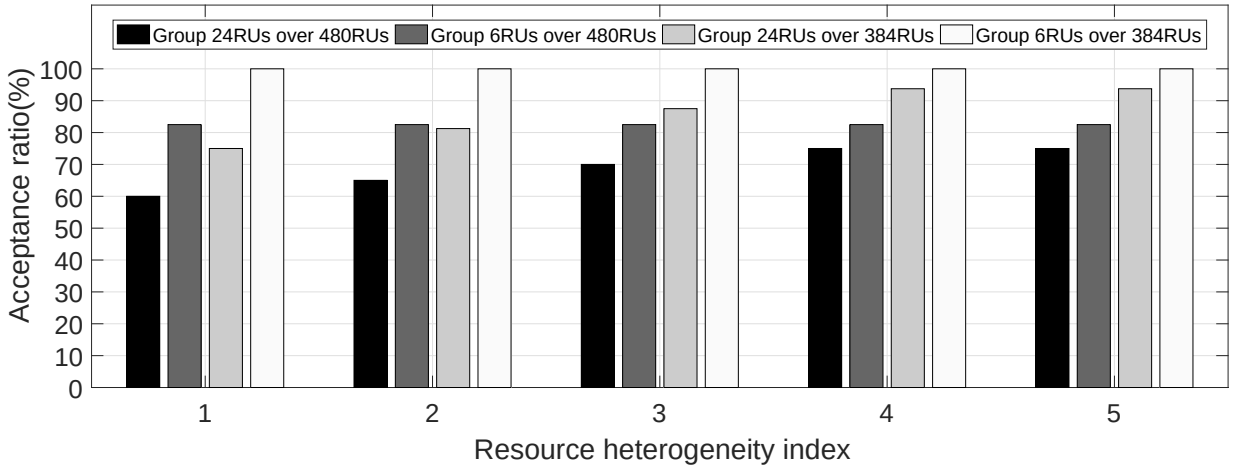
The acceptance ratios to place 384 chains for two RU group sizes are: 100% when grouping 6 RUs, and 75% when grouping 24 RUs for heterogeneous index 1, as shown in Fig. 6a. Such results justify the efficiency of our proposed approach and the higher acceptance ratio benefit when using a smaller RU group. When the node resources become heterogeneous, the orchestrator shall incorporate auto-scaling action to efficiently manage both infrastructure and slice workload dynamics. Here, Fig. 6a illustrates five different resource heterogeneity indexes and the corresponding acceptance ratios to place 384 and 480 service chains for two RU group sizes. When grouping 24 RUs, heterogeneity index 4 and 5 provide the largest enhancement as the DU functions (i.e., N_1 to N_6) consume more resources and better exploit the resource heterogeneity, while fewer enhancements are observed when RUs are grouped in 6.

To determine which action shall be taken to further increase the acceptance ratio, we compare the number of remaining resources at all DU nodes after placement (i.e., unused resources) and the number of unsatisfied requested resources (i.e., resources of rejected chains). From Fig. 6b, the remaining CPUs is more than the requested ones in the case of 384 RUs grouped in 24, which shall trigger a scale-up action to reallocate the unused resources to a subset of nodes to increase the acceptance ratio. In contrast, a scale-out action shall be triggered to provision more nodes in the case of 480 RUs grouped in 6 or 24, as the remaining resources are less than the requested ones.

To sum up, two strategies can be applied to enhance the acceptance ratio: (a) utilize a smaller group size of RUs as the minimum placement granularity, or (b) provision heterogeneous resources based on the service requirements to preserve scalability. The former requires adaptation when generating the service template, while the latter needs an agreement between the slice owner and the operator for the pricing model when different scaling actions are taken. However, the operator can update the shared service chain (e.g., change the functional splits) to optimize the resource utilization across different nodes, but it will impact every slice and so, shall be planned in a larger time-scale.

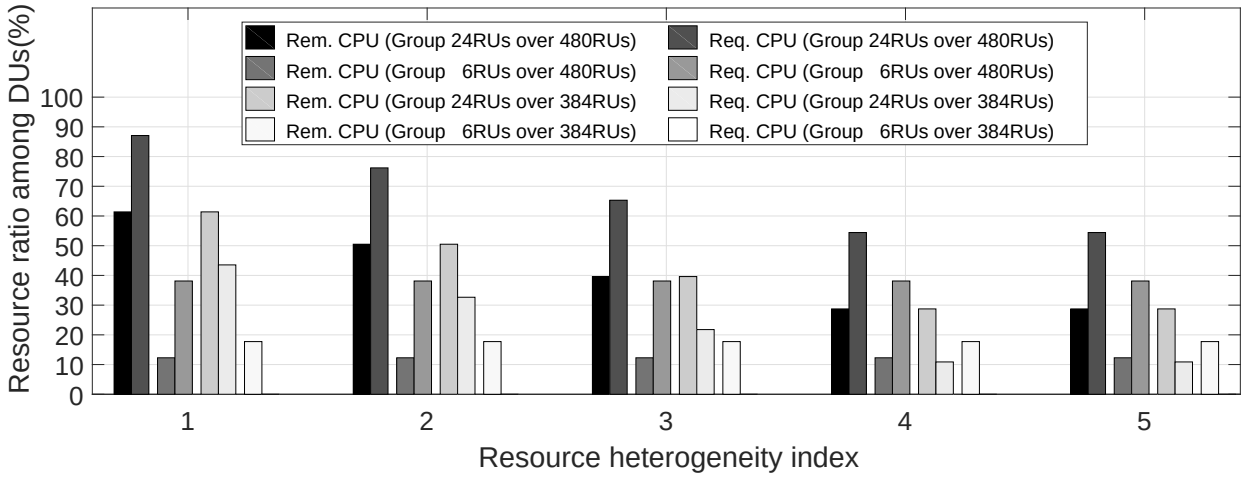
VI. Conclusions

We proposed a novel multi-service RAN runtime environment to provide a number of RAN runtime services, support slice orchestration and management, and enable flexible slice customization as per-tenant needs. We identified a number of new interfaces to enable the communication between orchestration and management systems with the RAN modules, through the RAN runtime and to facilitate the customized dedicated orchestration logic for each slice. We evaluated our approach in a disaggregated UDN deployment scenario for different levels of resource heterogeneity and showed the benefits of the auto-scaling mechanism to increase the service acceptance ratio.



Resource Heterogeneity Index	CPU resource													
	N_1	N_2	N_3	N_4	N_5	N_6	N_7	N_8	N_9	N_{10}	N_{11}	N_{12}	N_{13}	N_{14}
1	32	32	32	32	32	32	2	2	2	2	2	2	2	2
2	48	16	32	32	32	32	1	1	1	1	3	3	3	3
3	32	32	48	48	16	16	1	1	1	1	3	3	3	3
4	48	16	16	16	48	48	2	2	2	2	2	2	2	2
5	48	16	16	16	48	48	3	3	3	3	1	1	1	1

(a) Acceptance ratio



(b) Remaining and required CPU ratio

Fig. 6: Acceptance ratio and remaining/required CPU ratio for different resource heterogeneity indexes.

ACKNOWLEDGEMENT

This work has received funding from the European Unions Horizon 2020 Framework Programme under grant agreement No. 762057 (5G-PICTURE) and No. 761913 (SliceNet), as well as the French Government through the UCN@Sophia Labex ANR-11-LABX-0031-01.

REFERENCES

[1] M. Kamel, W. Hamouda, and A. Youssef, "Ultra-Dense Networks: A Survey," *IEEE Communications Surveys & Tutorials*, vol. 18, no. 4, pp. 2522–2545, 2016.

[2] A. Checko, H. L. Christiansen, Y. Yan, L. Scolari, G. Kardaras, M. S. Berger, and L. Dittmann, "Cloud RAN for Mobile Networks – A Technology Overview," *IEEE Communications Surveys & Tutorials*, vol. 17, no. 1, pp. 405–426, 2015.

[3] A. Ksentini and N. Nikaein, "Toward Enforcing Network Slicing on RAN: Flexibility and Resources Abstraction," *IEEE Communications Magazine*, vol. 55, no. 6, pp. 102–108, 2017.

[4] X. Foukas, M. Mahesh K., and K. Kontovasilis, "Orion: RAN Slicing for a Flexible and Cost-Effective Multi-Service Mobile Network Architecture," in *Proceedings of the 23rd Annual International Conference on Mobile Computing and Networking (MobiCom)*. ACM, 2017, pp. 427–441.

[5] NGMN Alliance, "Description of Network Slicing Concept," https://www.ngmn.org/fileadmin/user_upload/160113_Network_Slicing_v1_0.pdf, 2016, [Accessed on 13-Mar-2018].

[6] A. Nakao, P. Du, Y. Kiriha, F. Granelli, A. A. Gebremariam, T. Taleb, and M. Bagaa, "End-to-end network slicing for 5G mobile networks," *Journal of Information Processing*, vol. 25, pp. 153–163, 2017.

[7] A. Rostami, P. Ohlen, K. Wang, Z. Ghebretensae, B. Skubic, M. Santos, and A. Vidal, "Orchestration of RAN and Transport Networks for 5G: An SDN Approach," *IEEE Communications Magazine*, vol. 55, no. 4, pp. 64–70, 2017.

[8] P. Marsch, I. Da Silva, O. Bulakci, M. Tesanovic, S. E. El Ayoubi, T. Rosowski, A. Kaloxylos, and M. Boldi, "5G Radio Access Network Architecture: Design Guidelines and Key Considerations," *IEEE Communications Magazine*, vol. 54, no. 11, pp. 24–32, 2016.

[9] P. Rost, C. Mannweiler, D. S. Michalopoulos, C. Sartori, V. Sciancalepore, N. Sastry, O. Holland, S. Tayade, B. Han, D. Bega *et al.*, "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, 2017.

[10] V. Sciancalepore, K. Samdanis, X. Costa-Perez, D. Bega, M. Gramaglia, and A. Banchs, "Mobile Traffic Forecasting for Maximizing 5G Network Slicing Resource Utilization," in *Proceedings of 2017 IEEE Conference on Computer Communications (INFOCOM)*. IEEE, 2017, pp. 1–9.

[11] C.-Y. Chang, N. Nikaein, R. Knopp, T. Spyropoulos, and S. S. Kumar, "FlexCRAN: A Flexible Functional Split Framework over Ethernet Fronthaul in Cloud-RAN," in *Proceedings of Communications (ICC), 2017 IEEE International Conference on*. IEEE, 2017, pp. 1–7.

[12] K. Katsalis, N. Nikaein, E. Schiller, A. Ksentini, and T. Braun, "Network Slices toward 5G Communications: Slicing the LTE Network," *IEEE Communications Magazine*, vol. 55, no. 8, pp. 146–154, 2017.

[13] O. Arouk, N. Nikaein, and T. Turletti, "Multi-Objective Placement of Virtual Network Function Chains in 5G," in *Proceedings of Cloud Networking (CloudNet), 2017 IEEE 6th International Conference on*. IEEE, 2017, pp. 1–6.