Towards the quest for 5G Network Slicing

Farouk Messaoudi
IRT B<>Com
farouk.messaoudi@b-com.com

Philippe Bertin
Orange Labs
philippe.bertin@orange.com

Adlen Ksentini
Eurecom
adlen.ksentini@eurecom.fr

***Abstract*—Due to the recent mobile traffic explosion, Mobile network Operators (MNOs) are severely challenged. We believe that the imminent arrival of 5G, will drive changes in communications service provider networks and address several issues by leveraging on both Network Function Virtualization (NFV) and Software Defined Networking (SDN) technologies. The upcoming 5G ecosystem will involve a number of vertical markets to give rise to a plethora of novel services with different requirements such as, Ultra-Reliable Low-Latency Communication (URLLC), machine Massive Type Communications (mMTC), and enhanced Mobile Broadband (eMBB). To be able to offer such variety of services with different needs over the same network, this latter should be split into multiple logical networks known as network slices. In this paper, we propose an architecture design for 5G network slicing inspired by ONF and 3GPP models. We have developed an OpenAPI-based architecture that provides Network Slice as a Service (NSaaS) together with their life cycle management. We have demonstrated the realization of this approach via a deployment of the URLLC, mMTC, and eMBB services on a real platform.**

***Index Terms*—5G Slicing, NFV, SDN, Network Service.**

I. Introduction

The emerging 5G market is expected to bring a variety of services, allowing to meet the requirements of a highly mobile and connected society. The key enabler for 5G architecture is the support of a variety of vertical industries. In this context, the 5G Infrastructure Public Private Partnership (5G-PPP) working group [1] has distinguished *five* verticals; namely health-care, energy, media and entertainment, automotive, and manufacturing. These verticals will pave the way to a plethora of use cases and services in the industry sectors, pervasive human centric applications, and machine-type communications (such as e-health, connected cars, industry 4.0, 4K video streaming, and Virtual Reality (VR)).

The coexistence of these vertical industries depends on 5G network's ability to serve emerging services having different needs in terms of latency, bandwidth, reliability, capacity, and domain specific functionality. The *"one-size-fits-all"* networks architectural approach is unable to address such diverse Key Performance Indicators (KPIs) for all verticals. However, 5G networks with the current network softwarization trends (i.e., SDN [2] and NFV [3]), will leverage on programmability, flexibility, and modularity to create multiple logical networks. Each logical network, referred as *network slice*, is tailored for a specific service on top of mutualized network infrastructures. Separating the network into several logical ones' is not a new topic [4] (e.g., Virtual Private Network (VPN), Virtual Local Area Network (VLAN), and overlay networks), however, none of the existing solutions matches the expectations of 5G cellular services, with their requirements in terms of ultra-low latency, large bandwidth, and massive machine type communications.

Considering the need in revisiting networks architecture, design stage has already concentrated a lot of considerations. In [5], the authors have introduced the concept of NSaaS with service models and management. They have described three implementations of NSaaS; Core Network (CN) only, Radio Access Network (RAN) only, and both CN and RAN. Although the paper presents interesting background on NSaaS, it misses discussion on the deployment of the NSaaS through the ETSI NFV Management and Orchestration (MANO) framework. Network slicing applied to RAN has been designed in [6]. The authors have implemented a system for slicing wireless resources in a cellular network for RAN sharing among Mobile Virtual Network Operators (MVNOs) with a minimum impact on the access nodes design. The aim was to optimize the overall radio resource exploitation. Targeting similar objectives for efficient sharing of RAN among operators, in [7], the authors have designed RAN multi-tenant cell slice controller for flexible sharing of RAN resources among operators. A detailed discussion on RAN slicing has been presented in [8]; the authors have analyzed the RAN slicing problem in a multi-cell network in relation to Radio Resource Management that can be used as a support for splitting radio resources among RAN slices. They have compared four different RAN slicing approaches. Slicing the CN has been proposed in [9], wherein resources are divided according to traffic demand and reduce the CAPEX - OPEX. DECOR [10] is a 3^{rd} Generation Partnership Project (3GPP) approach for CN slicing, where dedicated 4G CNs are put together to meet the functional requirements of various sets of services. DECOR implicitly partitions the CN into slices by defining dedicated core elements, such as Mobility Management Entity (MME), for different services, running on dedicated and isolated hardware. In [11], the authors have discussed the feasibility to design a flexible and adaptive mobile CN based on functional decomposition and network slicing. Slicing the E2E network has been presented in [12], where the authors have developed a device triggered network control mechanism that allows 5G devices to discover, select, and access the most appropriate E2E network slices. Although several efforts have been provided on network slicing , most of them were focused on architecture design and simulations while they have not yet achieve real deployment.

In this paper, we unveil a novel network slicing architecture for integrated 5G communications we have designed, developed and experimented with our 4G/5G core network solution. Section II discusses enabling technologies. Section III highlights our proposed architecture. Section IV presents our experiment and the obtained performances, while section V draws some conclusions.

II. 5G Enabling Technologies

It is generally agreed that 5G network architecture will rely on NFV, SDN, as well as network slicing. Combined together, these technologies ensure flexibility, scalability, elasticity, programmability, and multi-tenancy support [13], [14]. To better understand our contributions in this paper, we introduce these technologies in the following.

A. NFV

NFV is an initiative to virtualize network functions and services (including, e.g. Packet Data Network Gateway (P-GW), Serving Gateway (S-GW), firewall, caching) traditionally run on proprietary, dedicated hardware. With NFV, Network Functions (NFs) are packaged as Virtual Machines (VMs) on commodity hardware, which accelerate the speed of time to market. NFV provides fair flexibility between the network's control plane and data plane through the scaling up and down of the allocated resources to meet the service demands [15]. NFV has introduced Virtual Network Functions (VNFs) concept, the software implementation of NFs, allowing the collocation of multiple VNFs instances on the same hardware or in the same cloud environment.

The European Telecommunications Standards Institute (ETSI) would ease instantiating vertical services by standardizing functional blocks and interfaces to describe the NFV architecture and orchestration framework [3], organized into three levels: VNFs, NFV Infrastructure (NFVI), and NFV MANO. NFV MANO is responsible for the orchestration and management of the NFVI resources on which it deploys Network Services (NSs) based on NFV. It aims to automatically instantiate and monitor NSs and manage their life cycle.

B. SDN

SDN aims to make networks agile, programmable, and support multi-tenancy [2], [16]. The key concept is to separate the network forwarding devices from entities that control. SDN is heavily promoted by the Open Networking Foundation (ONF) [17], which designed a slicing abstraction based on SDN [18], [17]. The architecture is modelled as a client-server relationship between an SDN controller and other entities. The SDN controller is at the center of the system architecture, autonomously enabling dynamic allocation, modification, and optimization of resources usage. When acting as a server, the SDN controller satisfies the client requests via the virtualization and orchestration of underlying resources. When acting as a client, it invokes services from other "underlying" servers. The SDN controller interacts with two main types of resource views: *client context* and *server context*. The client context represents the materials that the SDN controller needs to support a client's requirements. The server context represents the materials that the SDN controller needs to interact with a group of underlying resources. Resources could be a combination of network, storage, and compute nodes. The Operating and Business Support System (OSS/BSS) is responsible for the internal SDN controller configuration and administration, and both client and server contexts. The client context would have the same behavior with the 5G network slice as it provides the complete abstract set of resources and supporting control logic for constituting a slice. The resource group entity defines the semantics of the interfaces presented to the client. They can be virtual resources (e.g. infrastructure) or support resources, which are functions hosted in the SDN controller itself.

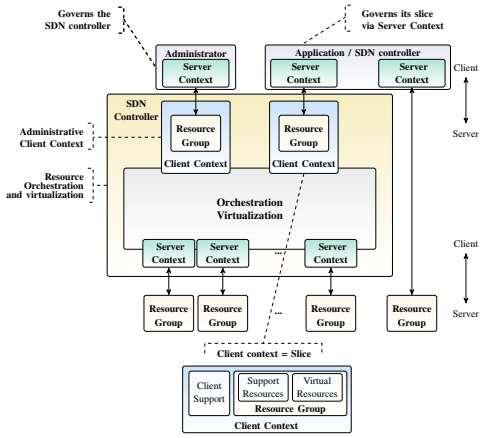


Fig. 1: SDN Architecture for Network Slicing

C. Network Slicing

The term "network slicing" has captured much attention within the research community and the industry, as well as Standards Development Organizations such as Next Generation Mobile Network (NGMN), 3GPP, and International Telecommunication Union - Telecommunication Standardization Sector (ITU-T). The NGMN considers the network slicing as the central part of 5G network architecture that relies on a 3-layer perspective; namely the resources, the network slice, and the service instances layer. The network slice instances are built with the combination of sub-network instances, eventually shared among multiple network slices. To describe this mapping, NGMN uses network slices blueprints (templates). On top of a network slice instance, multiple service instances are running. 3GPP defines the network slicing as a technology that "enables the operator to create networks, customized to provide optimized solutions for different market scenarios which demand diverse requirements, e.g. in terms of functionality, performance and isolation [19]". From the ITU-T perspective, the network slicing is seen as logical isolated network partitions composed of virtual resources, isolated and equipped with a programmable control and data plane [20].

III. Proposed Architecture

We describe in this section our baseline architecture for network slicing namely, the "*Vertical Slicer (VS)*", inspired by the ONF and 3GPP views, followed by the NSaaS proposal.

A. Vertical Slicer (VS) Architecture

The VS is a part of the OSS/BSS provider, it coordinates, via the Communication Service Management Function (CSMF), the requests for Communication/Vertical Services (CSs) composed from a set of vertical-oriented service blueprints. These blueprints are translated into Network Slice Templates (NSTs), which are mapped to Network Service Descriptors (NSDs) through the Network Slice Manager (NSM). The NSM allows the mapping of several NSDs into one network slice. Overall, the VS is responsible for the lifecycle management of both the CS instances and the corresponding network slices, as well as providing Authentication, Authorization, and Accounting (AAA) functionality as an entry point to the VS, and both Service Level Agreement (SLA) management and monitoring functionality.

Figure 2 shows the two main modules of the VS namely; the CSMF and the NSM, with a focus on the NSM. We were inspired by the ONF model description of the SDN controller (Figure 1). We used this logic for each 3GPP Management Function (MF) (i.e., CSMF, Network Slice Management Function (NSMF), and Network Slice Subnet Management Function (NSSMF)) [21]. Focusing on the NSMF, it is modeled as set of client-server relationship with the other MFs. In its role as a server, the NSMF may offer services (including, the exposure of NSTs, instantiating of network slices, and monitoring). While acting as a client, the NSMF invokes services from the NSSMF such as, the instantiating of network slice subnets. The architecture is organized into dual perspectives on the nature of client-server interfaces. The services (resource, respectively) perspective is appropriate from the *top-down* (*bottom-up*, respectively). The NSMF satisfy the client (i.e., CSMF) requests by its internal logic. We applied recursion of this architecture for each of the NSSMF and CSMF.

1) CSMF: It exposes to the verticals a catalogue of service blueprints. Internally, the CSMF keeps record of the received requirements per CS, and maintains a reference between these services and the Network Slice Instances (NSIs). For each CS, the CSMF performs accounting of the service requirements per NSI. The CSMF exposes two interfaces; the Northbound and Southbound interfaces (NBI and SBI, respectively). (i) The NBI towards the Verticals, provides blueprints and CS-related information (such as, discovery, life-cycle management, performance monitoring, and fault management). (ii) The SBI towards the NSMF, provides a catalogue of NSTs, selects the NST as a result of mapping between the CS requirements and the NST with the appropriate SLAs, requests for the allocation and the life-cycle management of the selected NSI. This function also provides via the SBI, the performance monitoring, fault management, and accounting, to verify if the SLAs are met.

2) NSM: Composed of NSMF and NSSMF. It is responsible for the management of both the NSIs and Network Slice Sub-net Instances (NSSIs), nested inside the NSIs. The NSM module is used to discover the NFs described in the ETSI NSDs. These NSDs are exposed to the NFV Orchestrator (NFVO) to build the NSTs. The NSM provides NSIs to the CSMF, which builds CSs on the top. In the case where the NSM provides directly the NSI to a vertical, this business service is called *NSaaS* (see Figure 3). The NSM keeps records of all the NSI requirements, it is unaware of the the CS instances but only their requirements received from the CSMF. When received, the requirements and/or SLA are translated into resource requirements for NFs to the ETSI NSs-Deployment Flavors (DFs). This process may go through an intermediate step, wherein the SLAs are mapped to VNF load estimations then, the VNF loads are translated into NSs-DFs. The VNF provider should provide a formula for the conversion of the SLAs to resource requirements. Beside the exposition, selection, and allocation of network slices, the NSM provides performance metering data of each NSI, events in NF failures or running below expectations, as well as monitoring and fault management services for each NSI.

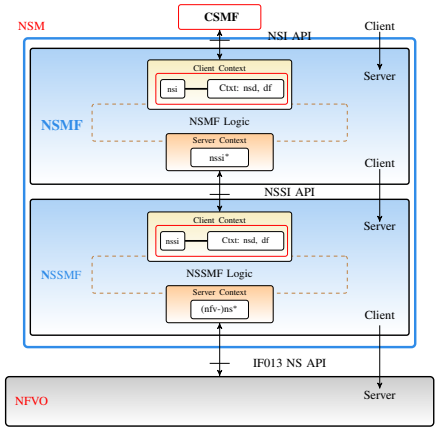


Fig. 2: (Simplified) view of the Vertical Slicer

B. NSaaS

A NSaaS is to provide a network as a service (which includes the RAN, CN), along with the services that it may support. This NSaaS is provided by the MVNO to its customers, leaving them the possibility to offer their own services on the top of the network slice. We distinguish three categories of business model for the NSaas: (i) *Business to Business (B2B)*, wherein the MVNO provides the network slice to a company who owns both the network and terminals and release to them the full control (eg., video surveillance networks for security companies and smart factory networks to manufacturing companies). (ii) *Business to Consumer (B2X)*- In this category, the end consumers are able to purchase customized data pipes from operators for their terminals such as smart home devices when using the Subscriber Identification Module (SIM) cards. In B2X model, the consumers generally do not

own the customized network, they just use it. (iii) *Business to Business to Consumer (B2B2X)*- Here the provider plays the role of wholesale provider to its consumers, while these latters play the role of providers themselves and sell their own services they built on the top of the network slice.

Figure 3 describes the architecture design of the NSaaS use case. The CSP via the NSMF exposes to the vertical limited network slice characteristics and management capabilities. Indeed, the vertical may negotiate with the NSaaS provider to have an exposure level for the NSI characteristics such as building new services like security functions, but also management features for example limiting the vertical operations to the commissioning/decommissioning of the NSI. The network slice exposes two interfaces; *customer facing* that support vertical (or business) services, and the *resource facing*, which supports the NFs (i.e., Physical Network Functions (PNFs) and/or VNFs) used to realize the Vertical Services (VSs). The categorization of a NSaaS is based on different criteria namely, the type of business service, the provided functionalities, location area coverage, capacity, QoS and, the SLA.

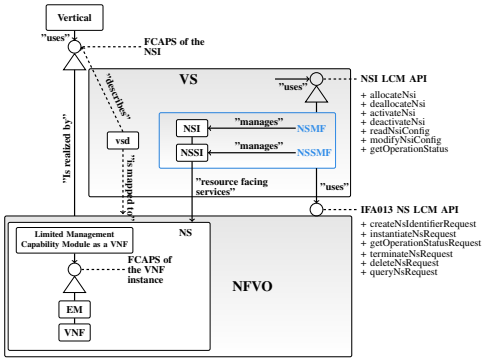


Fig. 3: Network Slice as a Service

C. Network Slice Creation Call-Flow

The workflow in Figure 4 describes the creation of a NSI, triggered by a vertical. As prerequisites, the blueprints, NSTs, NSDs, and VNF packages are already on-boarded, the vertical had chosen the appropriate blueprint, which is translated into a NST, which in turn is mapped to a NSD.

After the *nsd id* has been obtained via mapping, the *vsiLcm* requests the *nsiLcm* for the allocation of a NSI with the *nsd id* and *Input Params* (input parameters for the instantiating of the NS) as inputs and returns the *request status* and the *nsi id* **(1)**. The nsiLcm triggers the creation of the *nsi id* **(2)**. In **(3)**, the nsiLcm forwards the creation request to the *nsi*, which in turn requests the *Nfvo* for the creation of an *ns id* **(4)**. The Nfvo responds the nsi with the ns id **(5)**. After receiving the response from the Nfvo, the nsi records the mapping between (nsi id, ns id) **(6)**. In **(7)**, the nsi returns the *nsi id* to the nsilcm, which in turn returns it the vsiLcm **(8)**.

This workflow describes the creation of a NSI id, which needs to be activated. The workflow is not represented here

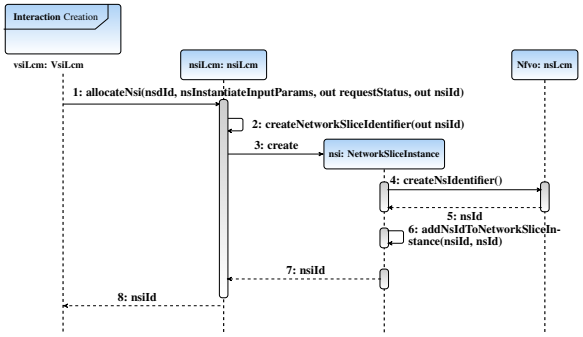


Fig. 4: NSI Creation Sequence Diagram

(due to the limited number of pages), however the logic of activation is as follows: The vsiLcm sends an activation request to the nsiLcm with the *nsi id* as input parameter and request status as well as *operation id* as results. The request is forwarded to the nsi, which requests the NfvoLcm for the instantiating of NS with the *ns id*, the one mapped to the *nsi id*. after the instantiating of the NS is done by the NfvoLcm, the *operation id* is returned.

D. Monitoring Heuristic

The aim of this heuristic is to monitor the instantiated NSs, and check if the SLA are met. As inputs of this heuristic, we have the SLA that includes the Quality of Service (QoS) (eg. latency), Quality of Experience (QoE) (eg. number of Frames Per Second (FPS)), and the number of sessions/second that the MME should support. As outputs, we compute the isolation ratio, SLA rates, and make a decision regarding the service and the SLA. The first action of the algorithm is to map the SLA to DFs, then compute r_x, θ_x, λ_x, and ω_x, which represent respectively, the network slice isolation ratio, the network latency satisfaction ratio, bandwidth satisfaction ratio, and the generated FPS satisfaction ratio. x corresponds to the service type (eMBB, mMTC, URLLC). α_{x,x_i} (β_{x,x_i}, respectively) represents the allocated (requested, respectively) resource of type x_i (radio, edge, transport, and central office) for the service x.

The purpose of this algorithm is to implement the concept of intent-based networking. Indeed, a *Service Assurance* component implemented in the VS (but not shown in Figure 2), maps the requirements of the top layer to the resources of the bottom layer (i.e., mapping of the SLA to DFs). Then, this component tries to satisfy the goals (here the SLA) and automate at scale.

IV. Performance

Our tesbed allows the "as a service" instantiation of several network slices over a single mutualized infrastructure. We use our 4G/5G mobile core solution which is fully virtualised and leverages on SDN to efficiently separate data plane from control plane features and traffic. This solution named the Wireless Edge Factory (WEF) is introduced in the following prior to the description of our complete testbed, use cases and experiment results.

Algorithm 1 Monitoring

Inputs: {SLA} : {Sessions/s, QoS}
Outputs: Isolation Ratio, SLA satisfaction rate, Decision
1. Map {SLA} to {DF}
2. Compute SLA ratios:

$$r_x = \left(\frac{\alpha_{x,x_1}}{\beta_{x,x_1}} + \frac{\alpha_{x,x_2}}{\beta_{x,x_1}} + \frac{\alpha_{x,x_3}}{\beta_{x,x_1}} + \frac{\alpha_{x,x_4}}{\beta_{x,x_1}}\right)/4$$
$$\theta_x = l_x^{\beta}/l_x^{\alpha}$$
$$\lambda_x = B_x^{\beta}/B_x^{\alpha}$$
$$\omega_x = (fps)_x^{\beta}/(fps)_x^{\alpha}$$

if $(\theta_x \geq 1)$ and $(\lambda_x \geq 1)$ and $(\omega_x \geq 1)$ **then**:
    The SLA are met
else
    if $(r_x < 1)$ **then**:
        Resources need to be scaled, Re-negociate the SLA
    else
        if $(r_x == 1)$ **then**:
            Perfect isolation and optimal usage of slices
        else NS can be scaled in/up
return r_x, θ_x, λ_x, ω_x

A. Wireless Edge Factory (WEF)

The WEF is a convergent, virtualized, SDN based, 4G/5G Core Network supporting multiple access technologies (including the Long Term Evolution (LTE), Wireless-Fidelity (Wi-Fi), Long Range (LoRa)). The WEF can be deployed at different locations in centralised Operator's or Cloud provider's data centers, distributed Point of Presences (PoPs) or even closer to the end-user. The SDN based separation between the control plane and the data plane brings the flexibility to host control plane VNFs in a centralised Cloud while data plane VNFs being distributed at (or closed to) each access site. It is hence foreseen that each access network will leverage on distributed data plane functions for efficient routing of users' traffic, while being controlled from a single control plane in the cloud. In our experiment, the WEF is instantiated in a network slice to (i) manage Wi-Fi and 4G access infrastructure built from standard equipment with multiple RAN access points per site (evolved NodeB (eNB) and Wi-Fi access point); (ii) unify subscribers management, authentication, IP addressing and security over the different technologies; (iii) provide efficient local users traffic switching policy capabilities thanks to the complete separation between the control plane and the user plane; (iv) be deploy-able as VNFs in off-the-shelf server.

Figure 5 depicts the WEF reference architecture. We rely on an Openstack centralised Cloud environment with a tenant dedicated to control plane VNFs (Home Subscriber Server (HSS), MME, AAA server, Dynamic Host Configuration Protocol (DHCP) server, S/P-GW Control Plane (S/P-GW-C), SDN controller), and a local server with KVM virtualization for data plane one's (S/P-GW User Plane (S/P-GW-U), DHCP relay, Network Address Translation (NAT)). Other components (4G RAN, UE, Wi-Fi AP) are based on commercially off the shelf products.

Please note that it is possible to instantiate data plane VNFs multiple times to create a complex topology network with several access networks.

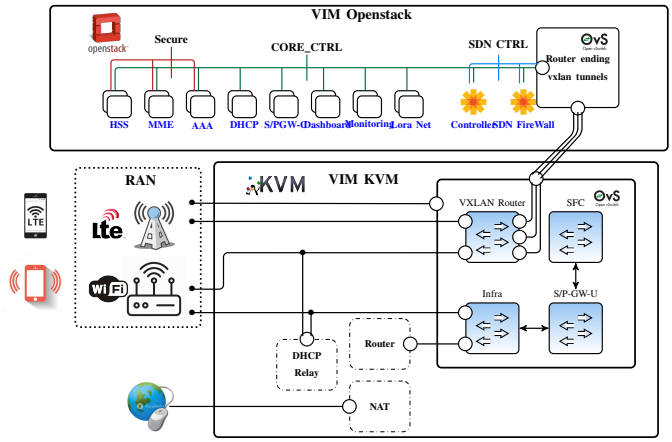


Fig. 5: WEF reference architecture

B. Testbed Description

b<>com "*Flexible Netlab*[1]" is a multi-tenancy dedicated environment. It is a private Cloud infrastructure for instantiating operational platforms for experimentation with 5G perspectives. In addition to Infrastructure as a Service (IaaS) and Platform as a Service (PaaS), it allows Testing as a Service (TaaS) with indoor and outdoor RANs using 4G LTE and 5G radio authorized by Arcep, WiFi on $2.4\,GHz$ and $5\,GHz$ bands, IoT LoRa in the $868\,MHz$ ISM band. The platform is capable of instantiating several 5G-capable infrastructure tenants, each one being autonomous. Flexible Netlab implements the MANO framework, that allows the instantiating and management of network services.

Figure 6 depicts the high level view of Flexible Natlab with a focus on the MANO framework. OSM (Release 4) is used as NFVO; OpenStack (Queens version) is the core Virtual Infrastructure Manager (VIM), which is backed by Ceph storage and connected to the Network Provider via L3 VPN. The network provides $2 \times 10\,Gbps$ aggregated datapath link; $10\,Gbps$ aggregated link for frontend & backend storage; and $1\,Gbps$ external link for interconnection with the provider.

C. Use Case: Wireless Hospital

John has recently been diagnosed with Cirrhosis 's disease. Due to an Hepatic insufficiency, John has the sharp decline in the transformation of nutrients and the elimination of toxic substances; he often feel tired, weak, and lose appetite. John is unable to do simple daily tasks. Even modest improvements in his digestive ability would greatly improve his quality of life and allow him to live autonomously. Waiting for a donator for years, he will finally receive a transplant from a new liver. John is transferred to the nearest hospital for an echography. The hospital environment such as temperature, parking and examination room availability, is supervised with

[1]https://b-com.com/sites/default/files/b-comFlexibleNetlab.pdf

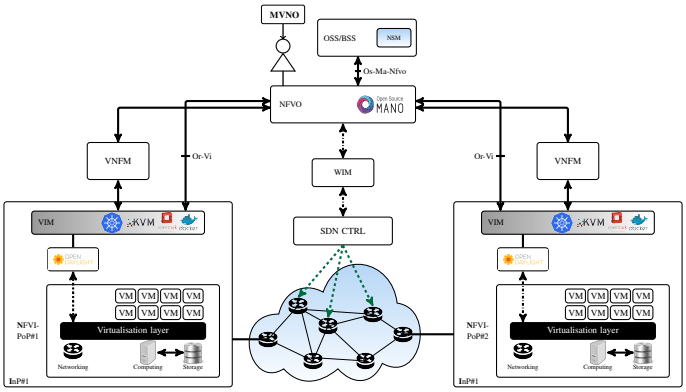


Fig. 6: The used platform for slices deployment

LoRa sensors. So, the ambulance is detected in the parking of the hospital and John is transferred in the first available room. A paramedic agent practices locally the echography via the remote supervision of a professor. The hospital relies on a telco operator who provides the control infrastructure to this MVNO. It has deployed several network slices including the mMTC, URLLC, and eMBB as depicted in Figure 7. The sensors belongs to the mMTC slice. The URRLC slice is used for echography streaming to the remote Professor. This service requires ultra-low latency, with relaxed constraints in terms of computing resources and bandwidth. Lastly, the eMBB, which needs large computing resources and high bandwidth, is assigned to the real time interaction between the remote professor and the paramedic agents (i.e., voice and video communications) through two HD webcam connected.

D. Results

We provide in the following our analysis from these experiments. We first deal with the SER and infrastructure cost KPIs for each service. Such an analysis is especially useful for MVNO, which aims to instantiate on-demand network slices quickly with a minimum cost. Then, we focus on performance of each service, which is the starting point for 5G communication services.

1) Service Instantiating Time (SER): Figures 8a depicts the ratio of needed time to instantiate each service (i.e., URLLC, mMTC, and eMBB) below x s in a population of 100 deployments. Two observations can be made in this graph:

- *No service has exceed* $240s$ *to be generated:* Thanks to service orchestration tools, which greatly reduce the time required to deploy and provision business services. Using orchestration makes overall operations much faster while also dramatically improves productivity.
- *The URLLC's SER is longer than both eMBB and mMTC's SER:* This is intuitive as the URLLC contains more VNFs than eMBB, which in turn contains also more VNFs than mMTC. Indeed, a service with a more VNFs takes longer time to be instantiated and interconnected.

2) Infrastructure Cost: Figure 8b illustrates the cost (in \$) generated from the allocation of resources to instantiate the three services. We have used two flavours; *c1r1, c2r2*. The c1r1 (c2r2, respectively) flavour corresponds to 1 (2, respectively) vCPU and 1 (2, respectively) GB of RAM. The service URLLC is composed of 10 VNFs; 9 VNFs with the c1r1 flavour ($9 \times c1r1$) and $1 \times c2r2$, while the eMBB (mMTC, respectively) service contains 8 VNFs; $7 \times c1r1$ and $1\times$ (2 VNFs; $2 \times c1r1$, respectively). The cost of $1 \times c1r1$ is 2.48\$ (5.83\$, 15.27\$, respectively) for large (medium, small, respectively) data-center. The costs in Figure 8b are obtained for multiple instantiating levels of the three services on the three data-center types, starting with the minimal one (10 VNFs, 8VNFs, and 2 VFNs for the URLLC, eMBB, and mMTC services respectively). At each instantiating level, we scale-up the service with one Virtual Deployment Unit (VDU). We note that the cost is proportional to the number of deployed VNFs as well as their flavours.

Now we move our attention to the URLLC service. We present in the following, the results of our measurement campaign regarding the SER for our use case as well as the infrastructure cost generated through the deployment of this service.

Figure 9 shows the Commulative Function Distribution (CDF) of the URLLC's SER KPI, which is obtained by increasing the flavours (in term of vCPU and RAM) for the MME. In our tests, at each instantiating level, we had increased the MME flavour. We used four flavours namely, c1r1, c2r2, c4r4, and c8r16, which correspond respectively to 1vCPU and 1GB RAM, 2vCPUs and 2GB Ram, 4vCPIs and 4GB RAM, and 8vCPUs with 16GB RAM. The results are showing that for the two first flavours (i.e., c1r1 and c2r2), we have almost the same SER, we believe that the allocation of resources with the flavour c2r2 is still small, and such resource are quit easy to find on the data-centers, which makes the SER for the service with this c2r2 flavour for the MME is similar to the one with c1r1 flavour for the MME. However, we notice a big difference between flavours c1r1 or c2r2 and c4r4 or c8r16, also between the c4r4 and c8r16. We notice that when the flavour is growing, the SER takes longer. We believe that this is due to the amount of vCPUs and RAM requested, which make resource less available on the data-center.

Figure 10 presents the time (in seconds) needed to instantiate the service URLLC (i.e. SER). We used box-plots as we want to focus on the variability of the SER. The aim is to quantify the stability of our framework. The box plot includes the 10^{th}, 25^{th}, median, 75^{th}, and 90^{th} percentiles of these times. We may notice three things here:

- SER is proportional to the MME flavour size: similarly to Figure 9, increasing the flavour size will increase the SER.
- High variability for high flavours: we remark that flavours c4r4 and c8r16 exhibit a high variability in the SER. This is due to the large resources that are needed to be instantiated. This is the worst case for the service provider, as this later cannot conclude if the service has encountered some issues or just because the instantiating takes longer. In this case, the provider needs to take the

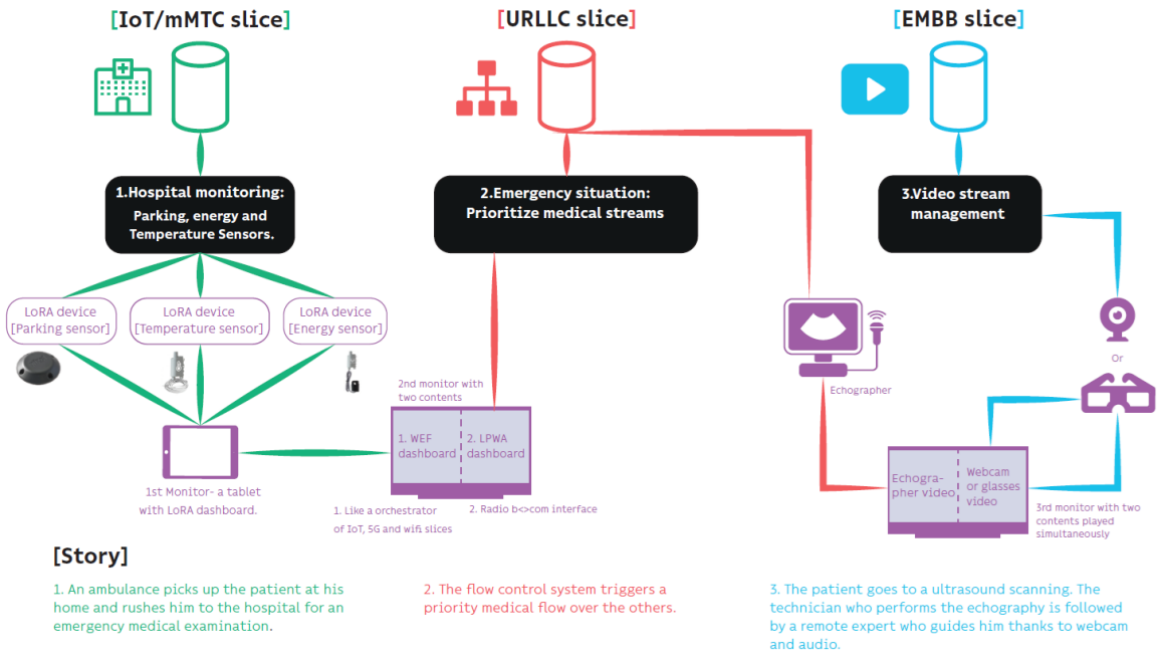
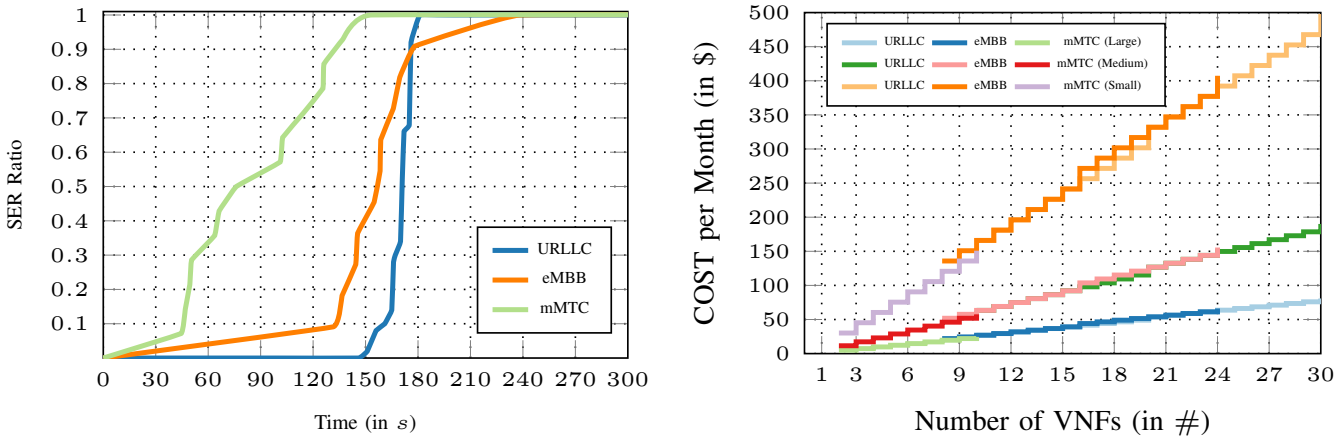


Fig. 7: Wireless Hospital use case



(a) SER

(b) COST

Fig. 8: **Calculated KPIs on Flexible Netlab**

decision of deleting the instantiating, which is not done yet or wait for more time. This high variability is clearly seen in c8r16 flavour, wherein the SER is about $485s$ for the 10^{th} percentile, and more than $785s$ for the 90^{th} percentile. The median is around $585s$. For the c4r4, the median is almost $500s$.

- Low variability for small flavours: this is the case of flavours c1r1 and c2r2. Where the 90^{th} and 10^{th} percentiles are so close that they nearly overlap, with a median around $171s$. This is ideal for service providers. Indeed, with this low variability, the provider will knows after a certain duration if the service is instantiated or not. It avoid the provider wasting time in waiting the instantiating of the service for long time, while the instantiating had issues. Therefore, after a certain duration waiting, the provider will clearly knows the re-instantiating of the service is needed.

Figure 11 shows the infrastructure Cost per month (in $) calculated from the deployment of the URLLC service versus the flavours chosen for the MME at each instantiating level. We are interested into the generated revenues for the infrastructure provider from allocating such a service. We observe that the infrastructure cost is increasing with the increasing flavours (i.e., increasing number of vCPU and RAM). Indeed, higher is the flavour size, more revenues will be generated. We also notice that the revenues for smaller data-centers are

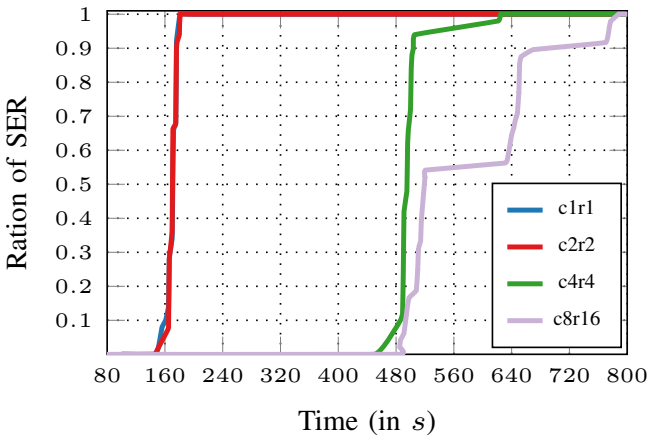


Fig. 9: CDF of SER for the URLLC service obtained by increasing the the MME flavour

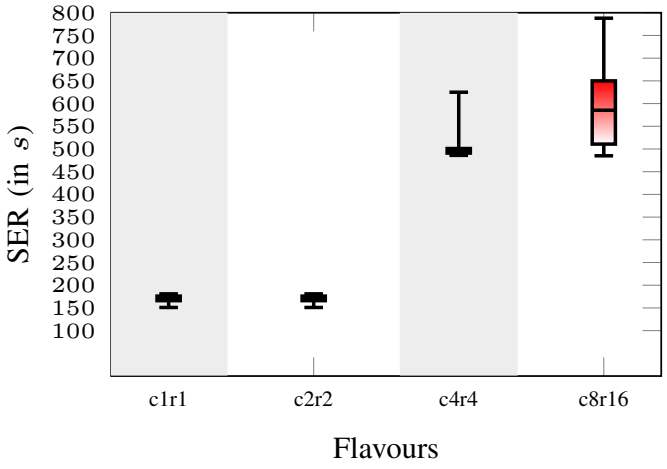


Fig. 10: SER for the URLLC service versus MME flavour. The box plot includes the 10^{th}, 25^{th}, median, 75^{th}, and 90^{th} percentiles of these times

higher than the revenues of larger data-centers. We believe that this is due to the scarcity of resources (vCPU and RAM), which is naturally more expensive to allocate than data-centers with resources that are supposed infinite.

We would like to know which of the two scaling approaches is consuming more: horizontal or vertical scaling (i.e., scale-up or scale-in). Figure 12 depicts the infrastructure Cost per month (in $) calculated from the deployment of the URLLC service according to the number of VDUs used for the MME at each instantiating level. The aim is to compare between the two approaches of scaling to see which of these solutions generates more revenues for the infrastructure provider. We may note three remarks from this figure.

- The cost is proportional to the number of VDUs. The generated cost for the infrastructure increases with the number of MME VDUs that are deployed. This is quite obvious, as the cost is an accumulating of the VNFs

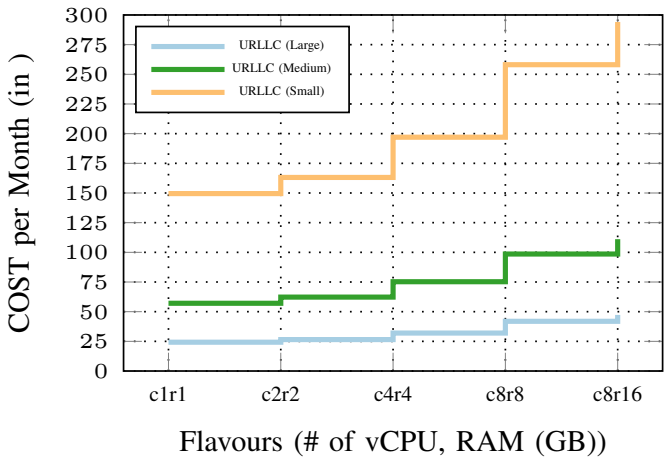


Fig. 11: Infrastructure cost per month generated from the deployment of the URLLC service over three platforms (large, medium and small data-centers), considering several flavours for the MME

flavour costs.

- The cost for small data-center is higher than the cost for medium data-center, which in turn higher than the one for large data-center. This is due to the scarcity of the resources. The more resources are rare, more will be high the infrastructure cost.
- The scale-up methodology is cheaper than the scale-in one. Indeed, from three VDUs for the MME, we can notice the difference in the price of the infrastructure. Cheaper is the cost for the scale-up, in which we increase the number of VDUs. We may explain this by the fact that flavours are chosen as power of two. That is to say, allocating 3 VDUs for the MME (therefore, we obtain 3 vCPUs, and 3GB RAM) in the scale-up method consists of three instantiating levels in the scale-in method (i.e., allocating the MME with the flavor c8r8, hence we obtain 8 vCPU, and 8 GB RAM).

Recent trends on how communication services are consumed have pushed for the need to revise the 4G network architecture. Indeed, mobile users are expecting to run more services with high QoS. The emerging 5G would represent the next step toward improving the user's experience. By consequence, generating QoS data would prove the need for 5G wireless technology, and help to understand how to move in this direction.

3) Network Latency: Figure 13a presents the latency of the URLLC service through a 4G network. The Round Trip Time (RTT) varies between $33ms$ and $59ms$ with a median at $48.2ms$, which is quit a lot for an URLLC services. However, with the upcoming 5G, which aims to drastically reduce this latency to $1ms$, this service will be feasible.

4) Bitrate: Figure 13b depicts the bitrate (in $Kbits/s$) for the URLLC and eMBB services in a duration of $60s$. We note that bitrate for the eMBB service has somehow repetitive

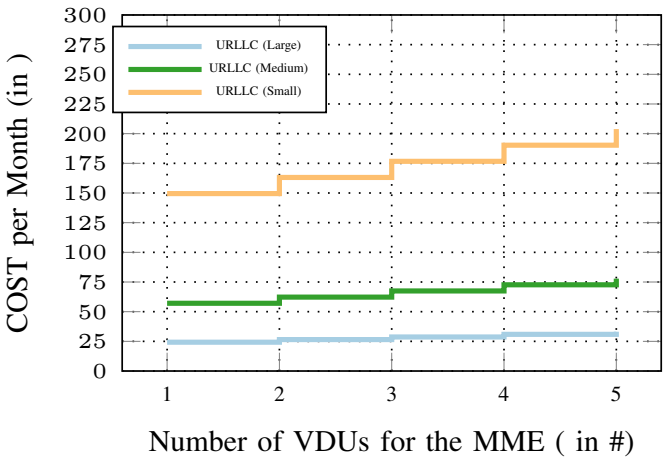
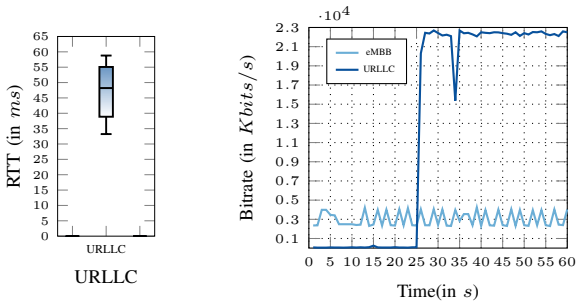


Fig. 12: Infrastructure cost per month generated from the deployment of the URLLC service over three platforms (large, medium, and small datacenters) considering the horizontal scaling (scale-in) for the MME



(a) Latency

(b) Bitrate

Fig. 13: **Slice Performance Metrics**

fluctuations. For the URLLC service, we note that the bitrate is almost 7.5× larger than the eMBB bitrate. Indeed, in the URLLC service, we use an echographer that capture data and stream them to a remote device. For responsibility issue, the stream is not compressed.

V. Conclusion and Perspectives

In this paper, we described a novel Network Slicing architecture for integrated 5G communications, featuring the NSaaS. We demonstrated its realization for the case of evolved LTE through a connected hospital use case, in which we deployed on-demand several network slices. We also provided an heuristic for monitoring of network services. Our work opens new exciting perspectives and research directions to improve the communication systems. Indeed, we may mention particularly a direction to follow for the network slices resource shortage. It is of interest to explore the NSD Deployment Flavor and Instantiation Levels of network services inside a network slice, and be able to automate the scaling and arbitrate the resource contention between NSIs based for example on priority management.

References

[1] 5G-PPP, "5g empowering vertical industries, white paper," 2016.

[2] W. Xia, Y. Wen, C. H. Foh, D. Niyato, and H. Xie, "A survey on software-defined networking," *IEEE Communications Surveys and Tutorials*, vol. 17, no. 1, pp. 27–51, 2015.

[3] ETSI GS NFV-MAN 001, "Network Functions Virtualisation (NFV); Management and Orchestration," 12 2014, version 1.1.1.

[4] N. M. M. K. Chowdhury and R. Boutaba, "A survey of network virtualization," *Computer Networks*, vol. 54, no. 5, pp. 862–876, 2010.

[5] X. Zhou, R. Li, T. Chen, and H. Zhang, "Network slicing as a service: enabling enterprises' own software-defined cellular networks," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 146–153, 2016.

[6] R. Kokku, R. Mahindra, H. Zhang, and S. Rangarajan, "Cellslice: Cellular wireless resource slicing for active RAN sharing," in *The 5th International Conference on Communication Systems and Networks (COMSNETS 2013), Bangalore, India, January 7-10*, 2013, pp. 1–10.

[7] P. C. Garces, X. Costa-Pérez, K. Samdanis, and A. Banchs, "RMSC: A cell slicing controller for virtualized multi-tenant mobile networks," in *The IEEE 81st Vehicular Technology Conference (VTC Spring 2015), Glasgow, UK, May 11-14*, 2015, pp. 1–6.

[8] O. Sallent, J. Pérez-Romero, R. Ferrús, and R. Agustí, "On radio access network slicing from a radio resource management perspective," *IEEE Wireless Communication*, vol. 24, no. 5, pp. 166–174, 2017.

[9] V. G. Nguyen and Y. H. Kim, "Slicing the next mobile packet core network," in *11th International Symposium on Wireless Communications Systems (ISWCS'14), Barcelona, Spain, August 26-29*, 2014, pp. 901–904.

[10] 3GPP TR 23.707, "Architecture Enhancements for Dedicated Core Networks," Tech. Rep. V13.0.0, 12 2014, release 13.

[11] M. R. Sama, X. An, Q. Wei, and S. Beker, "Reshaping the mobile core network via function decomposition and network slicing for the 5g era," in *IEEE Wireless Communications and Networking Conference Workshops (WCNC'16), Doha, Qatar, April 3-6*, 2016, pp. 90–96.

[12] X. An, C. Zhou, R. Trivisonno, R. Guerzoni, A. Kaloxylos, D. Soldani, and A. Hecker, "On end to end network slicing for 5g communication systems," *Transaction Emerging Telecommunications Technologies*, vol. 28, no. 4, 2017.

[13] T. Taleb, B. Mada, M. I. Corici, A. Nakao, and H. Flinck, "PERMIT: network slicing for personalized 5g mobile telecommunications," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 88–93, 2017.

[14] I. Afolabi, A. Ksentini, M. Bagaa, T. Taleb, M. Corici, and A. Nakao, "Towards 5g network slicing over multiple-domains," *IEICE Transactions*, vol. 100-B, no. 11, pp. 1992–2006, 2017.

[15] R. Mijumbi, J. Serrat, J. Gorricho, N. Bouten, F. D. Turck, and R. Boutaba, "Network function virtualization: State-of-the-art and research challenges," *IEEE Communications Surveys and Tutorials*, vol. 18, no. 1, pp. 236–262, 2016.

[16] Y. Zhang, L. Cui, W. Wang, and Y. Zhang, "A survey on software defined networking with multiple controllers," *Journal of Network and Computer Applications*, vol. 103, pp. 101–118, 2018.

[17] ONF TR-521, "Sdn architecture, issue 1.1," 2016.

[18] ONF TR-526, "Applying sdn architecture to 5g slicing, issue 1," 2016.

[19] 3GPP TR 23.799, "Study on on Architecture for Next Generation System," Tech. Rep., 12 2016, v14.0.0.

[20] ITU-T TR Y.3011, "Framework of Network Virtualization for Future Networks, Next Generation Networks-Future Networks," Tech. Rep., 01 2012, v14.0.0.

[21] 3GPP TS 28.530, "Management and orchestration; Concepts, use cases and requirements," Technical Specification (TS) 28.530, 12 2018, version 15.1.0.