Optimization of the Implementation of Network Slicing in 5G RAN

Nazih Salhab, Salah Elfalou, Rana Rahim, Salah Eddine Elayoubi, Rami Langar

Optimization of the Implementation of Network Slicing in 5G RAN

Nazih SALHAB
LIGM, UPEM, University Paris-Est, France
LTRM-EDST, Lebanese University, Lebanon
nazih.salhab@u-pem.fr

Salah EL FALOU
Faculty of Science
Lebanese University, Lebanon
salah.elfalou@ul.edu.lb

Rana RAHIM
Faculty of Science, LTRM-EDST
Lebanese University, Lebanon
rana.rahim@ul.edu.lb

Salah Eddine EL AYOUBI
Laboratoire des Signaux et Systèmes (L2S)
Centrale Supelec, France
salaheddine.elayoubi@centralesupelec.fr

Rami LANGAR
LIGM (UMR 8049 CNRS), UPEM
University Paris-Est, France
rami.langar@u-pem.fr

***Abstract*—Slicing is an emergent technology for 5G. It decomposes a single Radio Access Network (RAN) into multiple virtual networks "slices" to meet demands in term of throughput, mobility, latency, reliability, etc. Slicing needs real-time reconfigurations to keep current with demands' dynamics. This results in an increased cost of Operation Expenditures (OPEX). We approached this challenge as an optimization problem of infrastructure's resources. We virtualized and pooled Baseband Units (BBUs) resources on cloud. Dynamic allocation and interconnection with Remote Radio Heads (RRHs) are made possible by leveraging the advents of Network Function Virtualization (NFV) and Software-defined Networking (SDN). We implemented Distributed Base Station (DBS) using open software platform along to a public service orchestration tool for clouds. Our contribution is integrating service selection and deployment with real-time monitoring that allowed auto-control of resources by looping resources' lifecycle. In our experiments, we deployed several slices and we tested two scenarios. First scenario addressed slices' auto-scaling (Infrastructure Scale-Out/Scale-In) when free resources are available in the pool. Second scenario simulated slices' breathing (orchestration of resources) when the pool of resources is exhausted. In first scenario, results show that leveraging cloud elasticity by auto-scaling resources saves costs by providing exactly "what-is-needed" "when-it-is-needed" in term of cloud computing. In second scenario, results show that slices' breathing maximizes the usability by employing our "inhale-and-exhale" heuristic. It is about reusing resources from under-loaded slices in favor of overloaded ones with seamless effect on the user-experience.**

Keywords—5G; NFV; SDN; Network Slicing; Resources Auto-Scaling; Orchestration.

I. INTRODUCTION

5G is not just for Telecom Operators. It is for everybody. Unlike 4G that was mainly about technology enhancements 5G, planned to start on 2020, will not only be a throughput upgrade and enhancement of 4G. 5G will enable new applications by satisfying competing constraints such as latency, reliability, mobility, availability, energy efficiency, and throughput on top of varying connection and traffic densities. More than 70 use cases and scenarios of interests are identified for 5G by 3GPP [1]. Each use case needs a special mix in terms of these constraints that are to be optimized in a special context called Quality of Service (QoS) objectives.

Traditional Radio Access Network (RAN) architecture of mobile networks is not cloud-friendly, especially from scalability and reliability perspectives. This is resulting from the typical coupling of the control and data planes. Vendor lock-in resulting from the usage of dedicated hardware is an additional challenge. Software-defined Networking (SDN) decouples Control and Data planes [2] [3]. Network Function Virtualization (NFV) pools the infrastructure resources, namely "Compute, Network and Storage" and virtualizes them to offer Virtualized Network Functions (VNFs) [4] [5]. Cloud-RAN (C-RAN) is built upon Distributed Base Station (DBS) versions of Enode-B (ENB). It leverages the pooling of Baseband Units (BBUs) resources on the Cloud to seamlessly scale them, gain in statistical multiplexing and provide reliable back-end resources for Remote Radio Heads (RRHs) allocated at proximity of end-users [6]. The native elasticity of cloud allows auto-scaling of BBUs resources to handle the increasing demand of such resources [7].

5G is envisioned to answer different use cases by adopting slicing as a solution. Slicing provides optimized virtual networks (Slices) on a single air interface. Slicing offers different services.

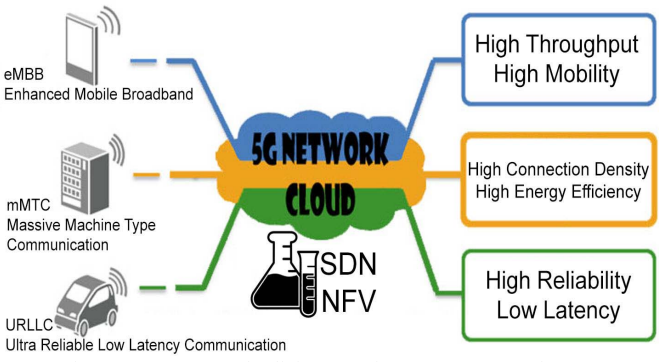

Figure 1 - 5G Network Slicing (Services versus Constraints)

They are "Enhanced Mobile Broadband" (eMBB) and two categories for Internet of Things that are "Massive Machine Type Communications" (mMTC), and "Ultra Reliable Low Latency Communications" (URLLC). "eMBB" is characterized by high throughput and high mobility. "mMTC" is characterized by high connection density and high energy efficiency. "URLLC" is characterized by high reliability and low latency [8]. Figure 1 depicts the concept of network slicing in 5G.

Repeated reconfigurations of resources allocation result in optimum resource allocation. However, this has a drawback of an increased cost of operation expenditures (OPEX) [9]. To keep cost affordable, automation of slices' resources optimization is needed. We approached the optimization of network slicing as resources optimization problem.

Resources Optimization is about executing and looping four stages of resources management processes starting by selecting and deploying resources, then monitoring and controlling them as depicted in Figure 2. In what follows, resources are used to denote any of the constraints binding a cloud computing system, namely, "Compute, Storage and Networking" resources. The resource optimization lifecycle starts with the planning process named as "Resource Selection". It is the first stage where adequate resources are identified for allocation. Next, in the execution process, "Resource Deployment" takes place. During monitoring and control processes, Resources are monitored to make sure that things are back in control through elicitation of feedbacks in order to make necessary adjustments by sending them back to the initial process. To do so, we virtualized the BBUs on cloud with constrained resources and used two algorithms to optimize allocated resources. Our contribution is to propose the automation of the optimization cycle aiming to save OPEX and cloud computing costs.

The rest of the paper is organized as follows. In section II, we give an overview of the related works. Section III describes the architecture of the system from a technical point of view. Section IV elaborates the algorithms for the studied two cases where only scale-out/in is allowed (Case I) or when no scaling is allowed but only orchestration is permitted (Case II). Section V discusses the performance evaluation, simulation assumptions and results. Section VI concludes the paper.

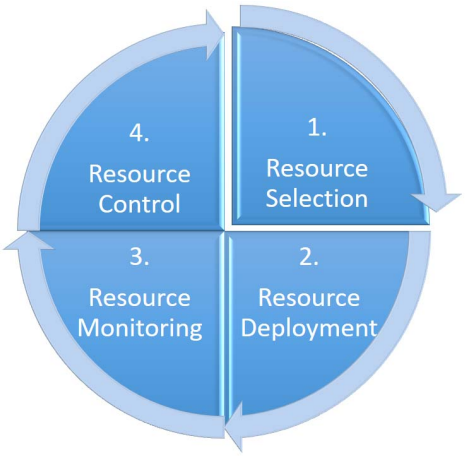

Figure 2 – Resource Optimization lifecycle

II. Related Works

Related works on optimizing slicing in NFV/SDN framework are limited. The authors in [10] have provided a prototype of software chaining using NFV. The authors in [11] have provided a demo for information centric networking using SDN for service chaining. The authors in [12] have presented an autonomic provisioning scenario in SDN Cloud Networks with NFV. However, all of these contributions provide particular use case that might not make optimum use of underlying resources and no detail on the algorithms were provided. The authors in [13] have implemented VNFs of C-RAN and made them publicly available at Juju Network Store allowing implementing the first stage of resource management (Resource Selection). However, this work relied on virtual machines, which consume bigger memory than containers. In addition, they did not provision a controller block allowing triggering collection of metrics on the performance of the VNFs to optimize the slices. Our contribution is using Juju charms for virtualizing the VNFs in Linux containers (LXC) rather than Virtual Machines (VM) and auto-controlling them. Once resources are deployed, coupling such resources with Real-Time monitoring service (Resource Monitoring) allows auto-control through an "Auto-scaler" to reach optimum slices (Resource Control).

III. Methodology

To model the three categories of services, namely EMBB, mMTC and URLLC envisioned for 5G RAN, a single host with limited resources is used. Consequently, the pool of resources is shared among the slices. Each category of service (EMBB, mMTC, URLLC) is implemented using an LXC.

Multiple levels of Orchestration are needed to provide optimum slicing and meet QoS objectives. Business orchestration serves as mapping of the application into corresponding Business Process (BP). Software orchestration maps the BP into the required services (slice). Platform orchestration shortlists and deploys the services onto the necessary network functions. Finally, infrastructure orchestration maps the platform onto adequate cloud infrastructure with a certain amount of resources expressed in terms of computing, storage and networking capabilities.

Figure 3 depicts the service-driven, layered architecture of the system. For example, Internet of Things (IoT) is mapped to either of URLLC or mMTC according to reliability and latency versus authentication complexity and power requirements.

The Platform layer works by deploying VNFs from Network Store (Juju Charm Store), interconnecting their interfaces and managing them with monitoring and controlling blocks. BBUs and RRHs are implemented using Eurecom-OpenAir-Interface (OAI-5G) [14], SDN Controller is implemented using FlexRAN Block [14]. HSS and EPC are implemented by OAI-CN and MySQL for AUC functionality [15]. Monitoring Service is implemented using the triplet (Telegraf for real-time metrics collector, InfluxDB for Time-series Database storage, and Grafana for Real-time web-based visualization) [16]. "Auto-scaler" is implemented using Elastisys Autoscaler [17].

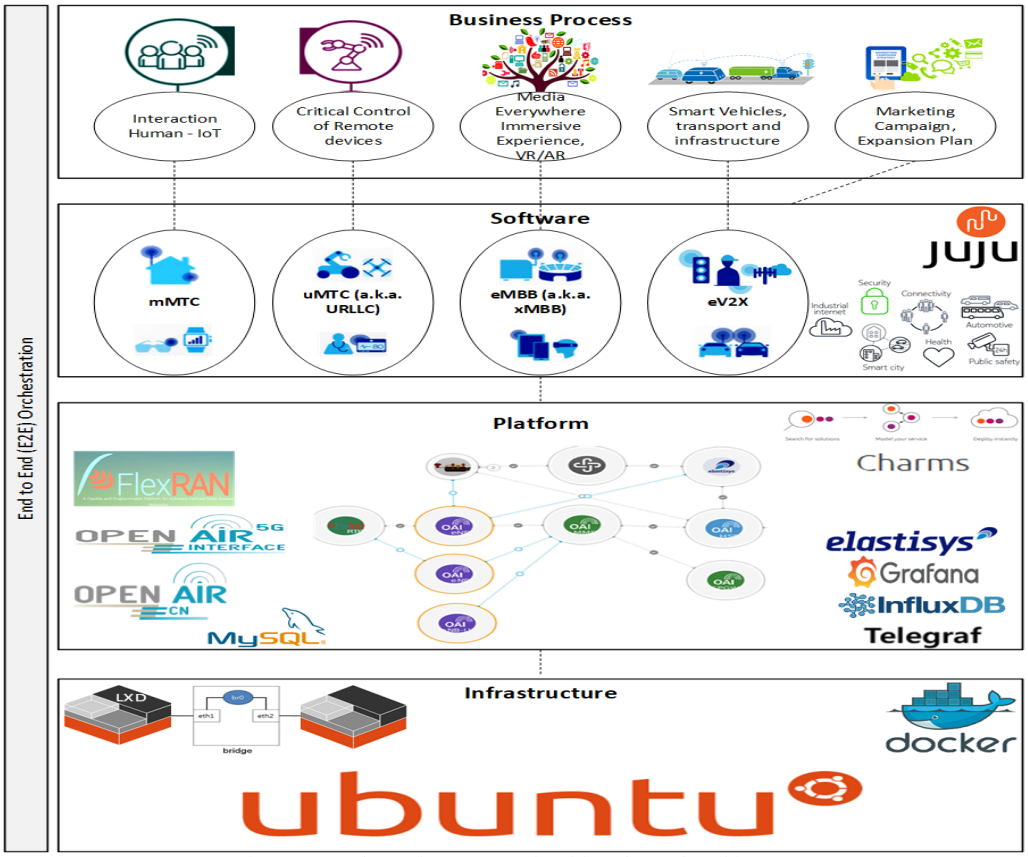

Figure 3 – Orchestration over Service driven layered Architecture

The bottom layer consists of the virtualized infrastructure and provides cloud based scalable resources. Usage of the cloud infrastructure allows implementation of C-RAN. LXD also known as "Linux container hypervisor" is a Linux daemon which provides a REST API to drive LXC containers. LXD is deployed as local cloud and is used for offline cloud computing along to a local bridge service to interconnect the containers. Selection and deployment of virtualized BBUs are done on offline Cloud (LXD over localhost) by using Juju Service Orchestrator offered by Canonical. It is used for the deployment of softwarized Network Element (NE) available at Juju online charm store to implement the VNFs of BBUs and RRHs [4]. Juju charms could also be deployed on many of the online clouds (Amazon AWS, Google GCP, Microsoft Azure, etc.) when additional computing power is requested. Docker is used as application container for the "auto-scaler" for faster instantiation.

In Figure 3, a slice is depicted in the platform layer as composed of an RRH and a BBU that are connected to the Evolved Packet Core Network (EPC) as known in LTE architecture.

IV. Algorithms Details

Although, SDN allows dynamic rerouting and reparenting of RRHs to BBUs. It might not be beneficial to do so in some scenarios. Such is the case where tighter coordination or mobility management constraints are imposed on the related BBUs. Consequently, when the BBUs resources are exhausted and there is additional demand for resources and provided that unused resources are available in the buffer, the preferred resort is to do scaling of the slice to cater the changing demands (Case I). Alternatively, when no scaling is permitted due to other constraint such as cloud computing cost or due to absence of additional idle resources, the resort is to make slice breathe. The "Auto-scaler" implements these algorithms in platform layer.

A. Case I: Cannot reparent RRH to another BBU; Mapping is not to be changed, slice needs to be scaled.

In some cases, there is a constraint of not reparenting any RRH to another BBU because the mapping is stipulated to be as-

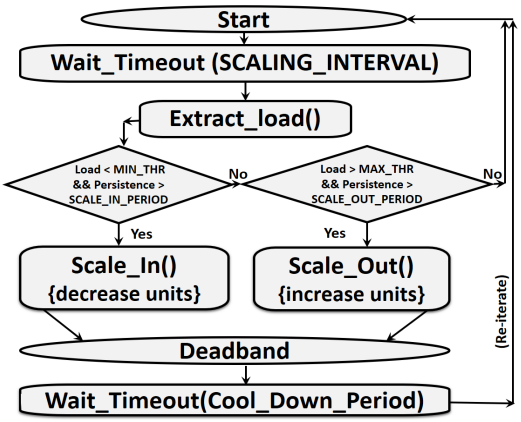

Figure 4 – Flowchart of the algorithm for "scaling-out/in" slices

Figure 5 – Case I - High load is relieved after scale-out decision.

is to minimize the inter-BBU handover. In such case, the only way to accommodate increased demand in terms of resources is to scale-out the BBUs to serve the requested load. Figure 4 depicts the algorithm managing the auto-scaling process. Conversely, in case of lower load than a predefined threshold, a scale-in event takes place to release the extra idle resources.

Consequently, such released resource will be made available in the buffer. Auto-scaling turns the risk of loss of sales, customer unwillingness to return also known as bad user experience and bad reputation into an opportunity of a greater exposure and a larger served user base.

This algorithm works as follows:

After a certain time laps equal to a predefined constant named "SCALING_INTERVAL", a function named "Extract_load" is called having as objective to collect the metrics about the real-time load of the slices. If the load is found to be lower than a certain threshold named "MIN_THR" and such low load persists for an interval of time that is bigger than a predefined parameter called "SCALE_IN_PERIOD", a decision to scale-in is taken to decrease the number of units in the pool to relieve the excess of resources. Similarly, for "MAX_THR", when the slice is overloaded and such state persists for a predefined interval called "SCALE_OUT_PERIOD", a decision to scale-out is taken to increase the number of units in the pool to provide room for handling the overload. Once a scale decision is taken, there should be a period of relaxation named "Deadband" that is forcibly entered immediately after a scale process. It is autonomously exited after expiry of a predefined timer called "Cool_Down_Period". Such relaxation is suggested in order to avoid snowball effect on the scale process and avoid having infinite loop when it comes to execution of the scale process.

For the "auto-scaling" algorithm, the choice of the "MAX_THR" and especially the "MIN_THR" might result in some unfairness. In fact, the "Scale-In" process, although it decreases the resources and cloud computing cost, when not well managed, it releases resources from the less-loaded pool and causes unfairness. By well-management, we hear the adequate choice of the thresholds "MIN_THR" and "MAX_THR". In order to insure fairness, a tradeoff between optimization and fairness is to be made. This is reflected in terms of values assigned to the thresholds.

B. *Case II: Can reparent RRH to another BBU; Mapping can be changed, Slice can breathe.*

The objective is to allow slices to "breathe" in a manner to offer additional resources to slices that are highly loaded. The aim is to remap some RRHs from the BBU that is highly loaded to another BBU that is less loaded. This is done by inhaling some resources from minimum loaded slice and exhaling the inhaled portion to overloaded slice progressively. This allows systematic decrease of the load on the overloaded slice until reaching the acceptable predefined threshold named "threshold_h".

The "inhale-and-exhale" algorithm is elaborated in Algorithm 1:

Algorithm 1 Breathe algorithm

```
1:  procedure Inhale-Exhale(slices)
2:    Initialize threshold_h;
3:    while (not all load(slice) <= threshold_h) do
4:      for each load(slice_i) > threshold_h do
5:         slice_j=find_min_load_min_degrad_slice();
6:         portion_i=offload(slice_i);
7:         slice_j=add_load(portion_i);
8:      end for
9:    end while
10: end procedure
```

V. Simulation Results and Discussion

In the simulations, we focused on only one of the resources' competing constraints (Compute, Network and Storage). It is the CPU load, which is primordial on the compute front. We found out that the two algorithms smooth the CPU load. This is an ultimate objective for Mobile Network Operators (MNO).

Accordingly, VNFs are supposed to run cost-efficiently and are expected to be responsive all the times, even when load spikes take place. This enhances the availability, saves cloud-

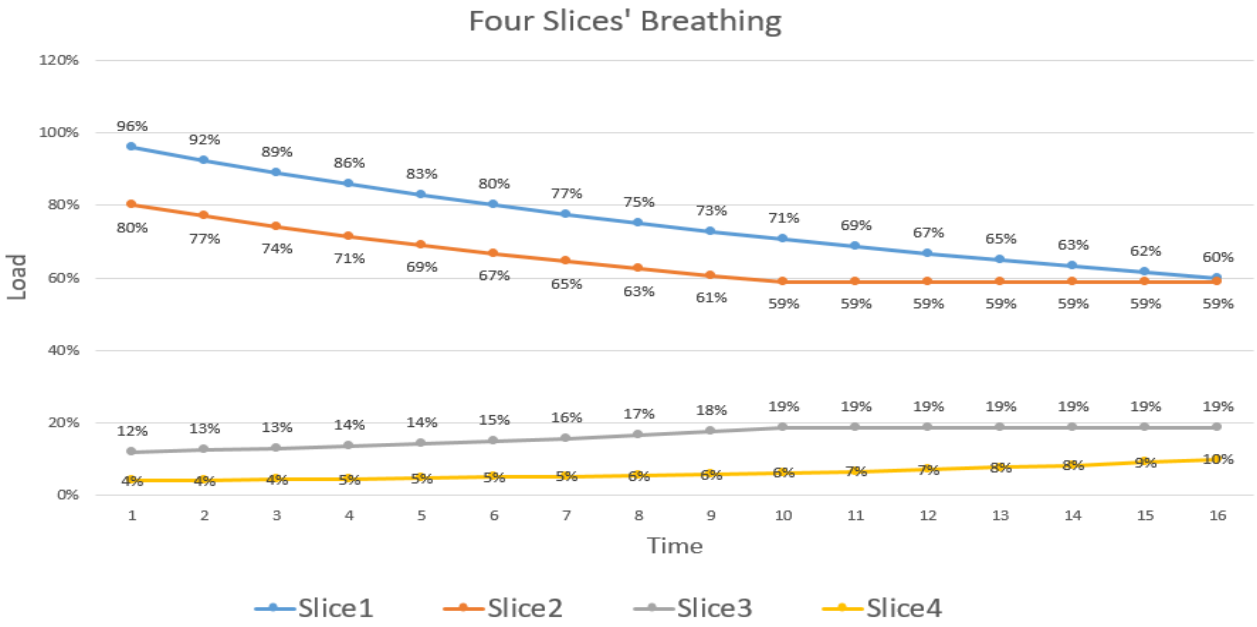

Figure 6 - Case II - Breathing, Inhaling resources from slices 3 and 4 and exhaling them to slices 1 and 2.

computing cost, and most importantly provides end-users with a better user-experience.

Case I:

Figure 5 depicts the status where there are already some idle resources in the BBUs. One of the slices is experiencing high-load. A scale-out decision is triggered to lessen the load on this overloaded slice. Indeed, the load is decreased once the new unit having new resources is up and running and starts sharing the load. The load of the other slice is barely affected which sustains the isolation principle [7]. The load is decreased from 80-90% to 40-50%, which is equivalent to 40% of load relieve. Note that in case of scale-out, the cost is increased. However, this is done efficiently to answer the needs and the demand for resources. Conversely, when scale-in takes place, the cost is decreased because of release of no-needed resources. This is an efficient usage of cloud computing resources aiming to insure optimum costs.

Case II:

No scaling is anticipated in this case. We relied only on resources' leveling followed by a smoothing as depicted in Figure 6. Breathing for all slices is done by borrowing some resources from the less loaded slices and lending them to the overloaded slices iteratively until reaching a predefined threshold that is set in this case as 60%.

The threshold is typically chosen in the range of 60% to 100% in a manner to maximize usage and avoid stressing the pool system. The choice of such threshold depends on the risk appetite and risk tolerance of the administrator. When chosen 100%, it is assumed that the cooling subsystem of the pool is perfectly working and no glitches in software or autonomous restarts are foreseen after an extended "fully-loaded" state. Otherwise, if no such guarantees are assured, when the administrator is risk-averse, such threshold is typically chosen less than 100%.

In our simulation depicted in Figure 6, "threshold_h" is chosen to be 60% (Risk Averse), and thus, loads of overloaded slice are progressively relieved through systematic decrease to ultimately have all slices loaded around the threshold "threshold_h". In this case, we do not anticipate a gain but just a load balancing resulting in a smoother operation of the slices.

VI. Conclusion

In this paper, we have addressed the optimization of slicing in 5G context using NFV/SDN and formulated it as an optimization problem of infrastructure's resources. The lifecycle started by selecting resources and deploying them in adequate platforms. Looping on resources' lifecycle while extracting metrics allowed monitoring and control of such resources to have optimum slices. We detailed the architecture used to experiment in 5G. We validated the two proposed algorithms. Our contribution resulted in significant enhancement of the resources' load front. As future work, we think of automating the fine-tuning of the values of the different parameters, namely: "MIN_THR", "MAX_THR", and "threshold_h" based on data learning. We think of automating the decision-making process on the choice of algorithm 1 or algorithm 2 using decision trees used in game theory. We will simulate an end-to-end system including dynamic users' arrival, and departure using a USRP SDR based prototype for C-RAN. We envision the design of algorithms for optimizing 5G slicing considering telecom constraints. Finally, new work on resources load levels prediction will be considered, so that a proactive action could be carried out to minimize the time of response to new demands.

Acknowledgement

This work was partially supported by the FUI SCORPION project (Grant no. 17/00464), "Azm & Saade Foundation", and Lebanese University.

REFERENCES

[1] 3GPP, "TR22.891, "FS_SMARTER", Feasibility Study on New Services and Markets Technology Enablers," 3GPP, 2015.

[2] J. Ordonez-Lucena, P. Ameigeiras, D. Lopez, J. J. Ramos-Munoz, J. Lorca and J. Folgueira, "Network Slicing for 5G with SDN/NFV: Concepts, Architectures, and Challenges," *IEEE Communications Magazine,* pp. 80-87, 2017.

[3] Open Networking Foundation, "TR-526 Applying SDN Architecture to 5G Slicing," Open Networking Foundation, California, 2016.

[4] ETSI, "Network Functions Virtualisation White Paper 5G," 21 February 2017. [Online]. Available: https://portal.etsi.org/nfv/nfv_white_paper_5g.pdf.

[5] ETSI, "Network Functions Virtualisation White Paper 3," 14 October 2014. [Online]. Available: https://portal.etsi.org/Portals/0/TBpages/NFV/Docs/NFV_White_Paper3.pdf.

[6] A. Checko, H. L. Christiansen, Y. Yany, 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, 2013.

[7] 5G Americas, "5G Americas White Paper – Network Slicing for 5G and Beyond," 5G Americas, 2016.

[8] I. d. Silva, G. Mildh, A. Kaloxylos, P. Spapis, E. Buracchini, A. Trogolo, G. Zimmermann and N. Bayer, "Impact of network slicing on 5G Radio Access Networks," in *Networks and Communications (EuCNC)*, Athens, Greece, 2016.

[9] R. Ranjan, B. Benatallah, S. Dustdar and M. P. Papazoglou, "Cloud Resource Orchestration Programming: Overview, Issues, and Directions," *IEEE Internet Computing,* pp. 46-56, 2015.

[10] J. Blendin, J. Rückert, N. Leymann, G. Schyguda and D. Hausheer, "Demo: Software-Defined Network Service Chaining," in *Software Defined Networks (EWSDN), 2014 Third European Workshop*, London, UK, 2014.

[11] M. Arumaithurai, J. Chen, E. Maiti, X. Fu and K. Ramakrishnan, "Prototype of an ICN based approach for flexible service chaining in SDN," in *Computer Communications Workshops (INFOCOM WKSHPS)*, Hong Kong, China, 2015.

[12] B. C. M. J. J. K. Z. M. M. M. D. Robert Cannistra, "Enabling autonomic provisioning in SDN cloud networks with NFV service chaining," in *Optical Fiber Communications Conference and Exhibition (OFC), 2014*, San Francisco, CA, USA, 2014.

[13] N. Nikaein, E. Schiller, R. Favraud, K. Katsalis, D. Stavropoulos, I. Alyafawi, Z. Zhao, T. Braun and T. Korakis, "Network Store: Exploring Slicing in Future 5G Networks," in *MobiArch '15 Proceedings of the 10th International Workshop on Mobility in the Evolving Internet Architecture*, Paris, 2015.

[14] 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.

[15] S. Costanzo, I. Fajjari, N. Aitsaadi and R. Langar, "A Network Slicing Prototype for a Flexible Cloud Radio Access Network," in *15th IEEE Annual Consumer Communications Networking Conference (CCNC)*, Las Vegas, NV, USA, 2018.

[16] M. Trojak, "Telegraf, InfluxDB, Grafana Docker Monitoring," Github, 2016. [Online]. Available: https://github.com/matisku/tig-stack.

[17] "Intelligent autopiloting for worry-free cloud operations," Elastisys, [Online]. Available: https://elastisys.com/.