Autonomous Network Slicing Prototype Using Machine-Learning-Based Forecasting for Radio Resources

Nazih Salhab, Rami Langar, Rana Rahim, Sylvain Cherrier, and Abdelkader Outtagarts

Abstract

With the emergence of virtualization and software automation for mobile networks, network slicing is enabling operators to dynamically provision network resources tuned to suit heterogeneous service requirements. This article investigates the architectures of the fifth generation (5G) of mobile networks experimental prototypes with a focus on network slicing. We present some existing 5G prototypes and identify their gaps. We then propose an architecture and a design of a 5G micro-service-based prototype. This prototype has the ability to auto-configure radio resources for network slices using machine-learning-powered decisions based on real-time acquired performance metrics. Finally, we discuss some use cases on top of this prototype and their related results before concluding.

Introduction

On top of conventional communication scenarios among people, the fifth generation (5G) of mobile communications and beyond (B5G) will empower machine-to-machine communication that drives different applications for multiple industries. Vertical applications with heterogeneous requirements in terms of latency, reliability, mobility support, energy efficiency, throughput, and connection/traffic densities will flourish with 5G/B5G. These applications include smart grid, collaborative robots, cooperative vehicles, smart farms, and so on. To serve such a plethora of applications, mobile network operators need to modernize their architectures to accommodate forecasted exponential increase in terms of number of connected devices and explosive size of exchanged data. Networks have to become agile by offering tailor-made infrastructures to suit specific use cases and their requirements. 5G standards development organizations envision supporting at least three major categories of services: enhanced mobile broadband (eMBB), massive machine-type communications (mMTCs), and ultra-reliable low-latency communications (URLLC). These are considered as network slices consisting of end-to-end logical infrastructures deployed over a shared physical infrastructure in line with performance demands. Network slicing and dynamic provisioning are especially interesting in the case of URLLC, where virtualized network functions have to be deployed at the edge of the network to minimize latency. Conversely, traditional system architectures rely on monolithic systems. In the monoliths, the presentation layer (user interface) and the data access layer (database) are tightly coupled in a single program, making the application self-contained and independent from other computing applications, but also highly dependent on the hosting platform. On the other hand, monoliths have some downsides, especially when it comes to the lack of flexibility in terms of maintenance complexity and evolution speed. With data usage growth, monoliths become too large to operate and maintain. In addition, in order to upgrade a functionality in a particular monolith, system administrators have to take the whole system down to complete such an operation. Moreover, a risk of regression during integration that causes unplanned delays should not be neglected. Therefore, traditional monoliths cannot provide required agility for 5G/B5G. The advent of network function virtualization (NFV) and software-defined networking (SDN) allow overcoming this limitation [1].

On another front, the cloud positions itself as an enabler for 5G/B5G and is well suited, in particular, for the radio access network (RAN) as cloud-RAN (C-RAN) with inherent elasticity. Instead of relying on dedicated servers and proprietary computing devices, the cloud provides virtual but dedicated resources relying on shared capabilities (compute, networking, and storage). Cloud resources provide required flexibility, featured through auto-scalability, and reduced deployment time, while ensuring up-to-date software releases [2].

In this context, we propose, in this article, a micro-service-based experimental prototype with machine learning (ML) forecasting capability for autonomous slicing management. Based on an exhaustive evaluation of multiple ML techniques [3], we implement the best performer prediction method and leverage our prototype to analyze its performance in a closed-loop setting. Also, compared to our former prototype [3], we consider a micro-service-based deployment of the core network rather than a monolithic one. Moreover, we have developed special scripts for the data processing engine and interfaced the latter with a

With the emergence of virtualization and software automation for mobile networks, network slicing is enabling operators to dynamically provision network resources tuned to suit heterogeneous service requirements. The authors investigate the architectures of the fifth generation (5G) of mobile network experimental prototypes with a focus on network slicing.

Nazih Salhab is with Amaris Consulting; Rami Langar and Sylvain Cherrier are with the University Gustave Eiffel; Rana Rahim is with the Lebanese University; Abdelkader Outtagarts is with Nokia Bell Labs.

Digital Object Identifier: 10.1109/MCOM.001.2000922

0163-6804/21/$25.00 © 2021 IEEE

We propose using a microservice based architecture that employs a collection of autonomous self-contained services, each implementing a single business capability. It is an evolution of a service-oriented architecture by further decomposing the services into smaller, independent microservices with loose coupling relation.

northbound application for autonomous configuration management.

The rest of this article is organized as follows. We first investigate the state of the art, the major European projects, and the standardized reference architecture for 5G. Then we present our approach in designing our 5G experimental prototype with ML capability using open source software. Finally, we present two use cases of the designed prototype using our Internet of Things (IoT) programming cloud platform, called Behavior Crowd Centric Composition (BeC3) [4] and our ML-based autonomous slicing prototype with forecasting capability, and discuss related experimental results.

Existing 5G Prototypes

The authors of [1] surveyed 5G network slicing using SDN and NFV. They discussed the related architectures and future challenges. They clearly advocated containerization technologies as an enabler for 5G. Moreover, the authors of [2] discussed the pros and cons of cloud migration and the points in between, suggesting the seven R's (replace, reuse, refactor, re-platform, re-host, retain, and retire) strategy, while emphasizing cloud architecture advantages. They addressed agnostic application migration issues. Therefore, we proposed a microservice-based cloud native architecture for our prototype using Docker [5].

The authors of [6] introduced the 5G-EmPOWER platform, which enables complex policies deployment and management over SDN. They evaluated the performance of their platform using Long Term Evolution (LTE) network elements provided by software radio systems (srsLTE).

The authors of [7] presented a prototype of a virtualized 5G infrastructure supporting network slicing. They investigated how OpenAirInterface (OAI) [8] can be used on top of Mobile Central Office Re-architected as a Datacenter (M-CORD). Moreover, they demonstrated a virtualized 5G RAN and core deployment using monolithic architecture. Recall that OAI is an open source software implementation of 4G/5G, spanning the full protocol stack of both radio and core networks. On the other hand, M-CORD is an open source reference solution for carriers deploying 5G mobile wireless networks.

The O-RAN [9] Alliance initiated an operator driven mission to open the RAN interfaces with the aim of allowing multivendor equipment to work seamlessly across well-defined interfaces. O-RAN aims to transform the RAN industry toward an open, intelligent, virtualized, and fully interoperable one.

MOSAIC-5G [10] encompassed multiple projects aiming to transform the RAN and the core into an agile service delivery platform to rapidly explore new ideas as well as emergent application and changing business needs. In particular, we reused the FlexRAN project [10, 11] to implement the RAN SDN controller and automate its control through our northbound web app for an autonomous configuration management.

The authors of [12] reported their experience building a 5G slicing prototype that enables multi-tenancy through virtualization. They highlighted the lack of monitoring capabilities, cross-domain and intelligent orchestration for such prototypes, and flagged these axes as open research questions. We note that these works did not include a holistic performance management stack or illustrate its benefits through common use cases. Therefore, we identify four objectives that a 5G prototype needs to address:
- Minimizing the total cost of ownership by relying on open source software
- Automating reconfiguration through data-driven ML decisions
- Overcoming IoT complexity through abstraction
- Minimizing human management by employing self-healing, resilience and autoscaling

Our proposed 5G prototype will indeed provide these features, as described later.

European Projects and Standards Development Organizations

The 5G Infrastructure Public Private Partnership (5G PPP) includes several projects aiming to formalize the reference architecture of 5G. Some of these projects are:
- 5G-NORMA (Novel Radio Multiservice adaptive network Architecture)
- 5G-xHAUL (5G-Cross Haul)
- COHERENT (Coordinated control and spectrum management for 5G heterogeneous radio access networks)
- Euro-5G
- Fantastic 5G (Flexible Air Interface for Scalable service delivery within the wireless communication networks of 5G)
- Flex5Gware (Flexible hardware/software platforms for 5G network elements and devices)
- METIS-II (Mobile and Wireless communications Enablers)
- 5G-ESSENCE (Edge Cloud computing and Small Cell as a Service).

On the other hand, several standards development organizations such as 3rd Generation Partnership Project (3GPP), European Telecommunications Standards Institute (ETSI) NFV Evolution and Ecosystem, Next Generation Mobile Networks (NGMN), TeleManagement Forum (TM Forum), and Open Networking Foundation (ONF) have envisioned a service-based architecture for 5G.

Architecture of Our Proposed Prototype

We propose using a microservice-based architecture that employs a collection of autonomous self-contained services, each implementing a single business capability. It is an evolution of a service-oriented architecture by further decomposing the services into smaller, independent microservices with loose coupling relation. A well-known example of this architecture is a cluster of microservices managed by an orchestrator. This architecture is convenient for large applications with numerous sub-domains that require high release velocity and high scalability. However, there are some downsides to this architecture: accrued latency due to distribution of microservices over several machines, and potential internal network congestion as more inter-service communication is involved. Note that such latency can be minimized by setting up a local repository to store the different network function images. Moreover, the high latency drawback appears when a virtualized network function is deployed from scratch. In contrast, when a scaling operation takes place, the change in terms of microservices replication happens seamlessly, without affecting the available microservices. In the following, we detail our proposed architecture for 5G experimental prototype as depicted in Fig. 1.

Infrastructure Layer (5G System)

To be in line with the reference 5G architecture, we deploy OpenAirInterface for Core-Network (OAI-CN) [8] as Docker containers in a Docker Swarm Cluster consisting of a manager and two workers. Note that we propose using Docker Swarm instead of Kubernetes [13] due to its native support by Docker, being the default orchestrator offered by Docker and thus much simpler when it comes to implementation in lab testing environments. Moreover, using containers allows us to manage the infrastructure in an agnostic way, allowing us to seamlessly upgrade it to different 5G releases whenever they are ready. Note also that we considered a number of replicas that is strictly greater than zero.

Accordingly, using the Docker Swarm container orchestration tool allowed us to seamlessly scale the number of replicas of each microservice without impacting the service to which they belong. Therefore, we have included in our auto-scaler a stopper-condition to enforce this in order to avoid any downtime. However, in the case of a failure of the hosting docker-machine, and supposing that all of the microservices are affected, it takes multiple seconds to redeploy the corresponding microservices. The downtime depends on multiple factors including loading it from the cached Docker images or having to download it from the online or local repositories. We also implement the next generation fronthaul interface to split the next generation Node-B (gNB) into:

- A remote unit implemented using a Universal Service Radio Peripheral (USRP) from ETTUS Research (a subsidiary of National Instruments)
- A digital unit (DU)
- A central unit (CU) using OAI-5G [8]

Note that OAI is an open source project that implements 3GPP-compliant technology on general-purpose x86 computing hardware and a USRP. Initially, OAI implemented the 4G architecture, but the OAI-wide community have started implementing 5G New Radio, the NF approach, and the control-user plane separation for the 5G Core (5GC). Recall that Docker [5] is a tool that enables developers to create, deploy, and run applications by using containers. Containers package up an application with all its dependencies (libraries and environment) and ship it out as only one package, thus bringing a vast economy of scale. Unlike virtual machines (VMs), containers virtualize an application on top of the operating system kernel directly, and thus containers are lighter to load, faster to start, and less memory-hungry when booting, compared to VMs.

Control Layer (SDN Controllers)

An orchestrator organizes containers at the networking level and enables the application to run as intended. We used Docker Swarm [5], the

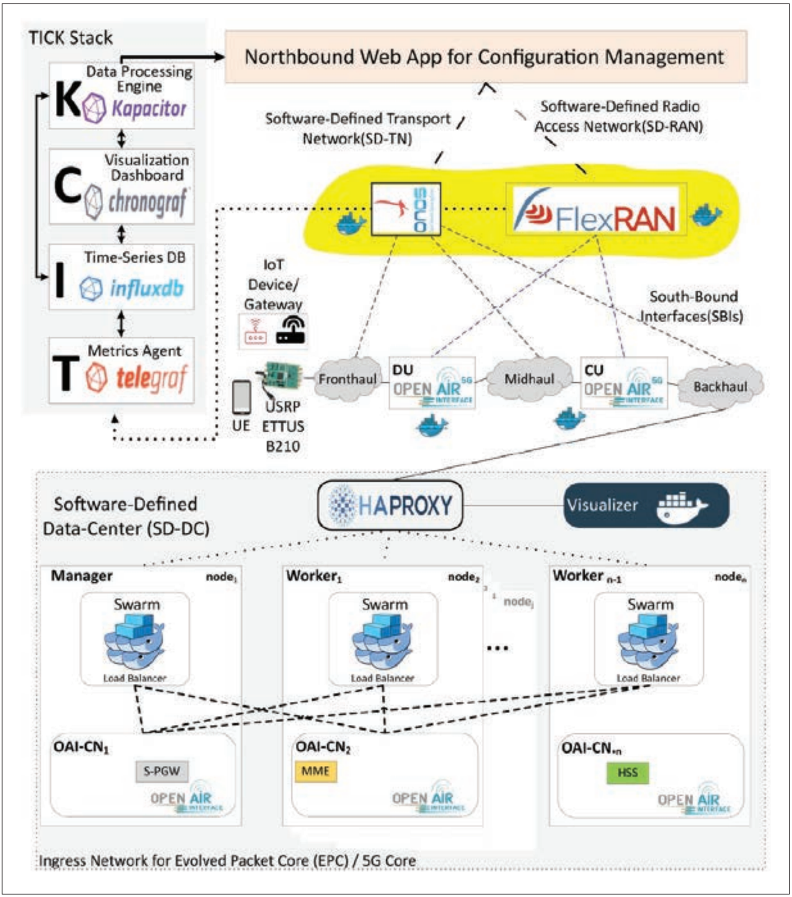

FIGURE 1. Experimental prototype architecture.

native Docker orchestrator, as it is well suited for small-scale deployments. In addition, it has a large community attached to it due to the huge number of developers that use Docker products. As controllers, we use FlexRAN [11] as a software-defined RAN controller (SD-RANC) to manage the gNB and open network operating system (ONOS) as a software-defined transport network controller to manage the fronthaul, midhaul, and backhaul networks.

Intelligent Application Layer (Top)

First, we have developed a northbound web app interfacing with the controllers for slices' life cycle management. It orchestrates the configuration management tasks, including on-demand slices creation/update/deletion, cell reconfiguration, devices rehoming to slices, and performance statistics generation. This layer enables us to create on the fly different slices as needed (eMBB/mMTC/URLLC). Based on our previous work where we exhaustively evaluated multiple ML techniques [3], we have shortlisted a particular regression model, as we see below, and integrated it to automate the provisioning of network slicing ratios. Furthermore, we have used BeC3 [4] for IoT applications to manage and control multiple IoT devices. BeC3 offers an intuitive interface that hides the complexity and specificities of IoT objects while giving unified access to their core functionality. Using this, we abstract the peculiarities of IoT devices in a virtual representation to get a reusable VM or a Docker image of such IoT devices. BeC3 allows implementing universal typical commands for IoT devices (e.g., ON/

In this work, we have focused on both eMBB and mMTC types of slices. Indeed, since wireless bandwidth is scarce and limited, eMBB slices are anticipated to be scaled in chunks on two fronts: bandwidth and service functions. On the other hand, massive sensors typically collect small-data payloads and exchange them with a base station.

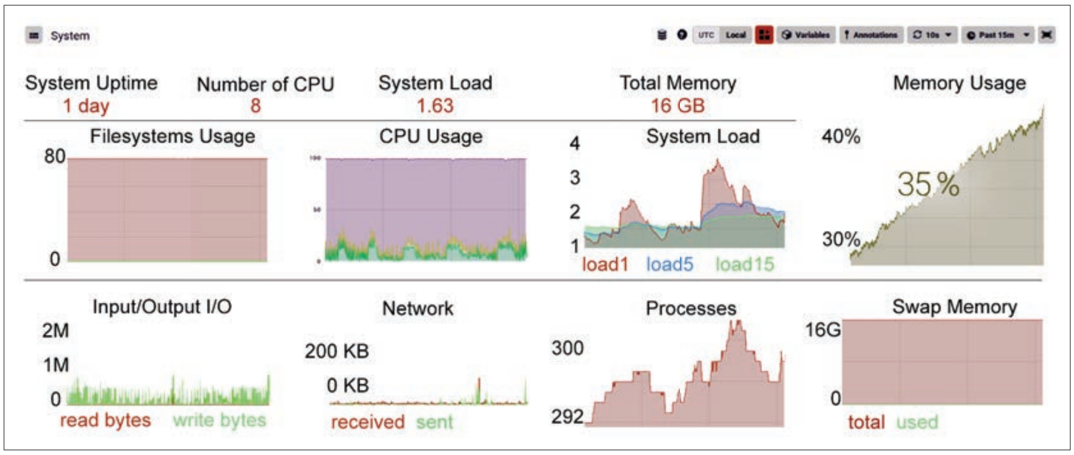

FIGURE 2. Physical machine metrics evolution over time.

OFF for smart switches, GET-TEMP for temperature sensor, GET-LUMINOSITY for light sensors). It also offers the possibility to control the behavior of integrated IoT devices, such as programming a device to upload measurement data to a remote server.

TICK Stack (Vertical)

In order to manage the performance of the virtualized network functions in real time, Telegraf, InfluxDB, Chronograf, and Kapacitor (TICK) [3] are all used as an end-to-end performance stack. Telegraf collects time-series data from a variety of sources, in particular, from the FlexRAN application programming interface (API) through JSN structured data describing the RAN metrics. InfluxDB delivers high-performance read/write access and efficiently stores fetched time-series data. Chronograf is used to visualize graphs of stored data in InfluxDB using different plot formats. Finally, we used Kapacitor to send commands to the northbound application. The TICK stack provides extract-transform-load capability from heterogeneous raw data and allows autonomous anomalies detection in time-series data.

Figure 2 depicts our hardware layer performance dashboard including system uptime, number of cores of the physical machine, system load, total memory, hard disk activity in terms of input/output (I/O) read/write, network activity, processes, and swap. All these metrics depict our physical host system capabilities (8 CPUs, 16 GB memory, etc.) when implementing our different use cases, shown in the next section. We can notice that the memory usage increases over time and reads 35 percent of the available memory due to the caching process.

Note that this prototype depicts a non-standalone deployment of 5G, but it could easily be upgraded to a standalone one or any other form of the eight envisioned deployment options of 5G. It is worth noting that our 5G experimental prototype takes into consideration self-healing, high-availability, centralized coordination, and horizontal scalability. Indeed, the swarm cluster for 5GC provides self-healing capability by re-instantiating a replica of a microservice whenever the latter fails due to intermittent software glitches or software/hardware failure of a worker machine and so on. To do so, it automatically recreates another microservice to compensate for the failed one and maintain the same requested number of microservices of the service in question. Using a cluster composed of a manager and multiple workers ensures high availability of the 5GC, meaning that if a worker fails, the cluster is maintained, and the remaining workers and manager absorb and balance the load using the internal or an external load balancer such as HAPROXY [14] (our case). The latter plays two roles: load balancing and reverse proxying. This service-based architecture ensures seamless scalability by means of horizontal scaling, specifically, by scaling-out or scaling-in the number of replicas according to changing needs in terms of computing resources. Note that the details of our auto-scaling algorithm can be found in our previous work [14].

Performance Evaluation

Dynamic Slicing and Devices Auto-Rehoming

In this work, we have focused on both eMBB and mMTC types of slices. Indeed, since wireless bandwidth is scarce and limited, eMBB slices are anticipated to be scaled in chunks on two fronts: bandwidth and service functions. On the other hand, massive sensors typically collect small-data payloads and exchange them with a base station. Due to the limited power of embedded sensors envisioned for mMTC slices, it is preferred that they work intermittently rather than continuously. Accordingly, the advantages of a dynamic slicing of radio resources are vivid in these two types of 5G network slices, as we see in the following.

It is worth noting that our proposed architecture could also address the URLLC network slices since our RAN network functions are deployed as microservices in a container-based virtual environment. According to our experiments [15], we have quantified the average deployment time of a container-based function to 1.8 μs ± 0.2 μs, which makes our prototype able to meet the critical time constraints of URLLC.

In this context, we have used our 5G experimental prototype [3] to manage several IoT devices including smart switches, smart lamps, and temperature/light sensors. We used our BeC3 IoT platform to provision the IoT devices at both the virtual and physical levels. Knowing that BeC3 is lightweight, we deployed it on a small device (Raspberry Pi 2 model B) to act as an IoT gateway connected to the 5G prototype either using Wi-Fi and tethered mobile Internet from a connected smartphone or through a USB dongle equipped with a corresponding SIM card.

We depict multiple use cases in Fig. 3. In particular, use case (A-*i*) illustrates the signal flow of a virtual Watt meter configured to send periodic measurements to the BeC3 cloud, where the results are displayed using a virtual display, both provisioned using BeC3. Use case (B-*i*) illustrates a manually triggered event using a smart switch to control a smart lamp through mobile Internet provided by our 5G experimental prototype. The smart devices use Enocean serial protocol 3 to transmit related radio telegrams. Finally, use case (C-*i*) shows an automated process of dynamic slice creation and a rehoming of IoT devices to an mMTC slice, while parenting smartphones to an eMBB slice.

The results of the dynamic slicing in the Chronograph dashboard can be seen in Fig. 4. For the sake of clarity, we chose a step of 40 percent of the available physical resource blocks (PRBs).

As the dynamicity of throughput requirements is more interesting in the case of the eMBB slice than that of the mMTC slice, we considered configuring dynamic auto-scaling of PRBs on the eMBB slice. We can see, in placeholder (2), that Slice 1, configured to use dynamic slicing, exploits the auto-scaling by increasing its ratio from 10 to 50 percent, then 90 percent, of available PRBs and vice versa. We can also see the corresponding allocated PRBs are (5, 25, and 45) out of the available 50 PRBs of our setup, as reflected in placeholder (9), which depicts the instant downlink PRBs. Remaining placeholders depict the downlink/uplink percentages for both slices 0 and 1, in addition to the instant Power-HeadRoom (PHR) and the downlink volume, which increases along the usage.

Slicing Ratio Forecasting Using Machine Learning

Using our proposed prototype, we initially acquired considerable RAN data to train multiple ML models. In particular, we have selected regression trees as our preferred ML forecasting mechanism as we have found it to be the best performer in terms of root mean square error (RMSE), mean average error (MAE), and coefficient of determination (R2) [3].

Recall that a regression tree is built using binary recursive partitioning, which is an iterative process that splits the data into partitions or branches. Then it continues splitting each partition into smaller groups as the mechanism moves up through each branch.

Using our prototype, and specifically by carrying out a query to the database (InfluxDB) tables, we extracted the historical traffic profile including the timestamp and the PRBs used by each connected device. We then aggregated the used PRBs per type of slice. We also enriched the traffic profile by appending the requirements from the slice owner. All of these constituted our set of predictor variables. On the other hand, according to the implemented infrastructure usage policy (relying on iterative auto-scaling), we appended

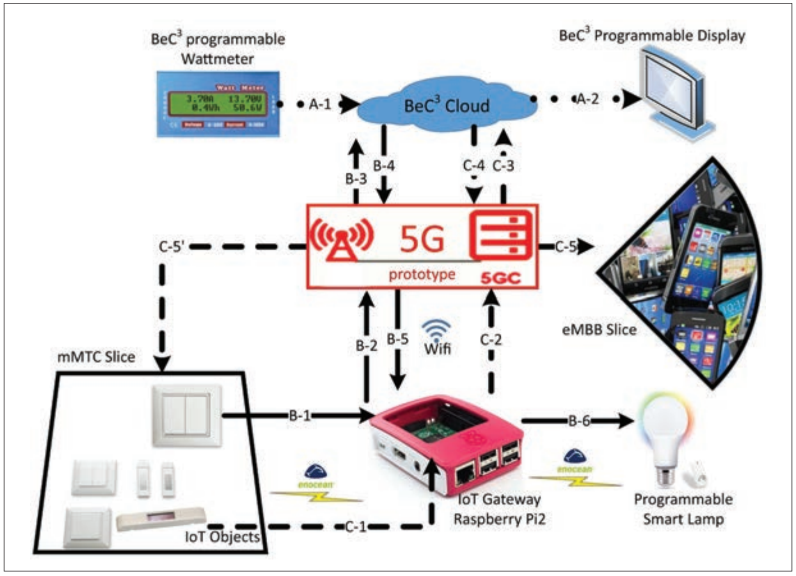

FIGURE 3. Use cases for evaluation of our prototype.

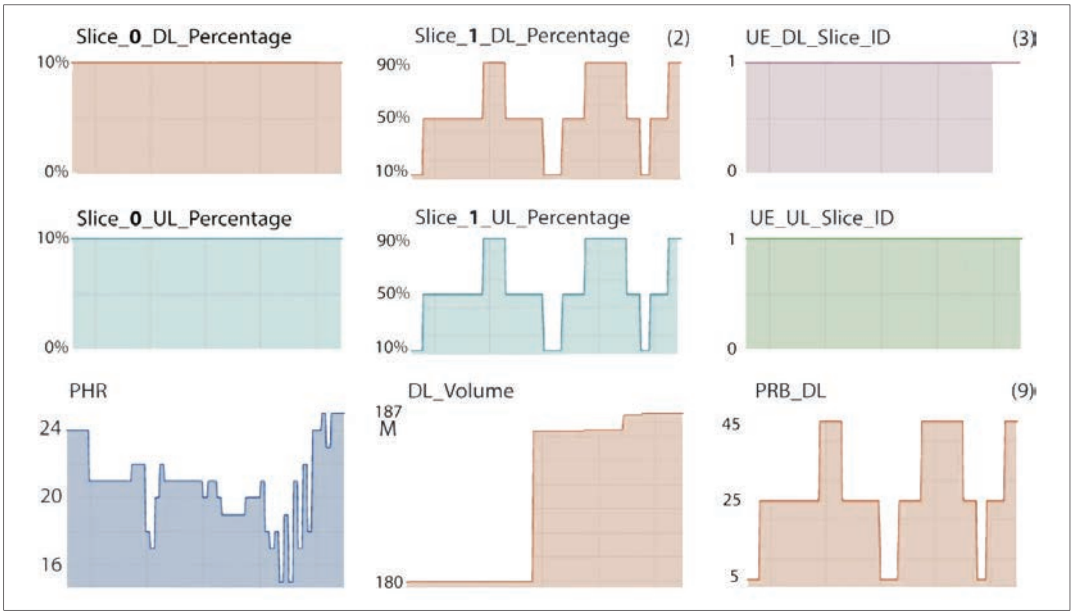

FIGURE 4. Radio Metrics Dashboard over Time.

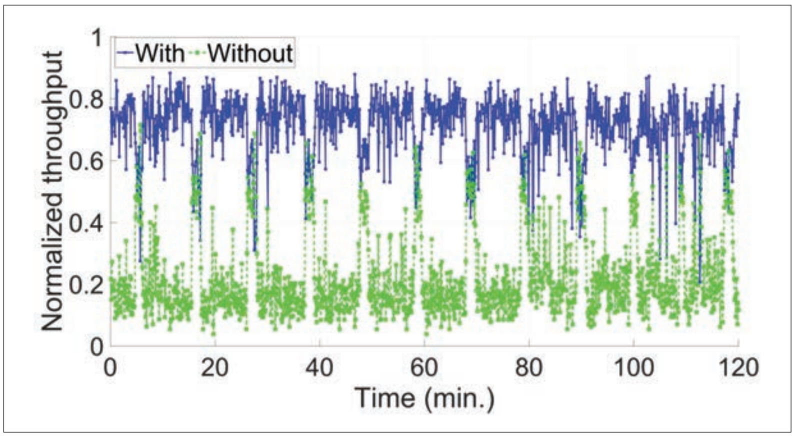

FIGURE 5. Normalized throughput with/without slicing ratios forecasting.

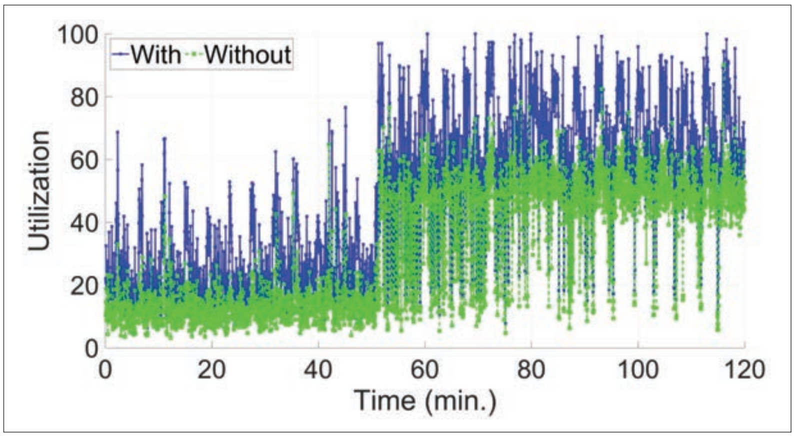

FIGURE 6. CPU utilization with/without slicing ratios forecasting.

to our enriched traffic profile the related response for each slice in terms of allocated PRBs. Accordingly, we fed this whole table to the regression tree learner.

We implemented, in the northbound web application, our ML-based forecasting mechanism to orchestrate slicing ratios for multiple network slices based on available PRBs. Once the training of our ML model is done, our objective would be to dynamically provision the optimum slicing ratio out of the available pool of PRBs to the new created slices. To do so, we start by creating one slice hosting a smartphone and map it to an eMBB slice. We use the Youtube video streaming application to simulate some bursty traffic. Note that this slice is configured to use our dynamic slicing through forecasting, whereas a second slice is dynamically created whenever the IoT devices have to upload some data streams. This latter slice will then be automatically deleted after finishing the transfer process, and its related PRBs will be returned to the PRB pool. Accordingly, the remaining active slices can use them when needed. We evaluated the system performance using two metrics, namely the normalized throughput and the CPU utilization of the host machine.

Figure 5 depicts the normalized throughput for the eMBB slice during two hours of simulation when using our autonomous network slicing vs. the case where we deactivate it. The normalized throughput is computed as the ratio between the obtained throughput and the maximum available one. We can easily observe that the forecasting process allowed increasing the throughput by approximately 30 percent compared to the case where the forecasting is deactivated and only a static slicing ratio of 50 percent to each of these two slices is used. We notice that forecasting slicing ratios accelerates the provisioning of required PRBs to serve the requirements of the running applications.

On the other hand, we can see that this automation comes at the cost of increased utilization of CPU of the host system, as depicted in Fig. 6. This is explained by the increase of computing resource utilization resulting from the processing of the slice life cycle (creation/destruction of a slice). Finally, we note that the same normalized throughput is maintained, although we queued a new video, starting from 50 minutes and onward. This is seen in the system utilization, where the trends of CPU usage for both cases (with/without slicing ratios forecasting) increased, reflecting such additional load.

Conclusion

In this article, we investigate the available prototypes for 5G and propose a novel architecture. Our architecture differs from existing state-of-the-art monolithic ones as it relies on a microservice-based implementation. We also complement it with a TICK Stack for performance monitoring and management. On the application layer, we developed two web applications for configuration management, leveraging machine learning forecasting techniques to auto-provision slicing ratios. We propose to manage the IoT devices using BeC3. Using Docker Swarm, we implement self-healing, high availability, centralized coordination, and horizontal scaling. We validated our implementation using a realistic use case with IoT sensors so that we dynamically create a slice and rehome devices to it. We observe that using a machine-learning forecasting model has substantially increased the throughput of the network, based on the same radio network design, with the drawback of increased computing resources utilization.

Acknowledgments

This work was supported by FUI SCORPION project (Grant no. 17/00464), Paris Ile-de-France Region (Grant no. 19CNE00437), "Azm & Saade Foundation," and the Lebanese University.

References

[1] A. A. Barakabitze *et al.*, "5G Network Slicing Using SDN and NFV: A Survey of Taxonomy, Architectures and Future Challenges," *Computer Networks*, vol. 167, 2020, p. 106,984.

[2] D. S. Linthicum, "Cloud-Native Applications and Cloud Migration: The Good, the Bad, and the Points Between," *IEEE Cloud Computing*, vol. 4, no. 5, 2017, pp. 12–14.

[3] N. Salhab *et al.*, "Machine Learning Based Resource Orchestration for 5G Network Slices," *Proc. 2019 IEEE GLOBECOM*, 2019, pp. 1–6.

[4] S. Cherrier *et al.*, "BeC3: Behavior Crowd Centric Composition for IoT Applications," *Mobile Networks and Applications*, vol. 19, no. 1, 2014, pp. 18–32.

[5] Docker; https://www.docker.com/, accessed: Jan. 10, 2021.

[6] E. Coronado, S. N. Khan, and R. Riggio, "5G-EmPOWER: A Software-Defined Networking Platform for 5G Radio Access Networks," *IEEE Trans. Network and Service Management*, vol. 16, no. 2, 2019, pp. 715–28.

[7] C.-Y. Huang *et al.*, "Design and Prototype of a Virtualized 5G Infrastructure Supporting Network Slicing," *Proc. 2018 IEEE 23rd Int'l. Conf. Digital Signal Processing*, 2018, pp. 1–5.

[8] OpenAirInterface; https://openairinterface.org/, accessed Jan. 10, 2021.
[9] O-RAN; https://www.o-ran.org/, accessed Jan. 10, 2021.
[10] MOSAIC-5G; https://mosaic5g.io/, accessed Jan. 10, 2021.
[11] X. Foukas *et al.*, "Flexran: A Flexible and Programmable Platform for Software-Defined Radio Access Networks," *Proc. 12th Int'l. Conf. Emerging Networking Experiments and Technologies*, 2016, pp. 427–41.
[12] X. Foukas *et al.*, "Experience Building a Prototype 5G Testbed," *Proc. Wksp. Experimentation and Measurements in 5G*, 2018, pp. 13–18.
[13] Kubernetes; https://kubernetes.io/, accessed Jan. 10, 2021.
[14] N. Salhab, R. Rahim, and R. Langar, "NFV Orchestration Platform for 5G over On-the-Fly Provisioned Infrastructure," *Proc. IEEE INFOCOM Wksps. 2019*, 2019, pp. 971–72.
[15] S. Matoussi *et al.*, "5G RAN: Functional Split Orchestration Optimization," *IEEE JSAC*, vol. 38, no. 7, 2020, pp. 1448–63.

Biographies

Nazih Salhab (nazih.salhab@univ-eiffel.fr) is a subject matter expert on mobile networks working at Amaris Consulting. He received his dual Ph.D. from University Paris-Est, France, and the Lebanese University. Before that, he received his Master's and Engineer's degrees from the Lebanese University. He has been leading business analysis, operation, and project management for mobile network operators for more than 15 years. His research interests are 5G/6G, network slicing, C-RAN, SDN, NFV, and machine learning.

Rami Langar (rami.langar@univ-eiffel.fr) is currently a full professor at University Gustave Eiffel (UGE), France. He received his Ph.D. degree in network and computer science from Telecom ParisTech, France, in 2006. His research interests include resource management, network slicing in 5G/6G systems, C-RAN, SDN, machine learning, and mobile cloud offloading.

Rana Rahim (rana.rahim@ul.edu.lb) is an associate professor at the Lebanese University. She received her Ph.D. degree in January 2008 from the University of Technology of Troyes, France. Her research interests include 5G/6G, QoS, IoT, smart grids, C-RAN, network slicing, and SDN.

Sylvain Cherrier (sylvain.cherrier@univ-eiffel.fr) is an associate professor at UGE. He received his Ph.D. degree from University Paris-Est, France. His research interests include Internet of Things management, choreography, and automation.

Abdelkader Outtagarts (abdelkader.outtagarts@nokia-bell-labs.com) is a senior researcher and leader at Nokia Bell Labs. He received his M.Sc. and Ph.D. degrees in automation of energy systems from INSA Lyon, France, and an M.Sc. degree in software engineering from the ETS of Montreal. His research interests include 5G/6G, NFV, SDN, and machine learning.