The case for serverless mobile networking

Marco Gramaglia
University Carlos III of Madrid
Madrid, Spain
mgramagl@it.uc3m.es

Pablo Serrano
University Carlos III of Madrid
Madrid, Spain
pablo@it.uc3m.es

Albert Banchs
University Carlos III of Madrid
IMDEA Networks Institute
Madrid, Spain
banchs@it.uc3m.es

Gines Garcia-Aviles
University Carlos III of Madrid
Madrid, Spain
gigarcia@it.uc3m.es

Andres Garcia-Saavedra
NEC Laboratories Europe
Heidelberg, Germany
andres.garcia.saavedra@neclab.eu

Ramon Perez
Telcaria Ideas S.L.
University Carlos III of Madrid
Madrid, Spain
ramon.perez@telcaria.com

***Abstract*—The softwarization of communication networks provides notable benefits, such as flexibility, improved resource efficiency, and commoditization. In exchange, softwarization requires an increased management overhead and the need to re-design network operation. While the mobile networking ecosystem is currently adapting this new paradigm with other network-related aspects (e.g., network slicing), cloud computing already addressed such problems with the introduction of *serverless* architectures, also known as Function as a Service (FaaS). With this approach, the software is decomposed into its minimum building blocks, i.e., functions, maximizing scalability, resource efficiency, and flexibility. In this paper, we analyze the potential adoption of the FaaS paradigm by the mobile networking ecosystem, discussing the implicit advantages, the challenges to address, and some solutions to overcome them.**

***Index Terms*—Network Slicing, Network Softwarization, Serverless computing**

I. Introduction

The fifth-generation (5G) of mobile networks will tackle the current and future trends of data consumption, especially from mobile devices, while fulfilling stringent requirements on delay, reliability, and throughput, among others. To provide customized services that efficiently meet these requirements, mobile networking is adopting two key technologies from computer science, namely, *softwarization* and *modularization*. In a nutshell, the first is the ability to operate fully-fledged networks through software components, while the second consists of defining and instantiating re-usable and highly focused Virtual Network Functions (VNF). Thanks to these, network providers can move away from highly specialized hardware solutions and benefit from building deployments based on general-purpose hardware architecture running re-usable software components.

While the adoption of softwarization and modularization requires a non-negligible cost, e.g., the management overhead or the re-design of certain functions that now run as software components instead of as hardware implementations [1], in exchange it provides significant benefits, such as enabling the *network slicing* paradigm [2]. Apart from these benefits, another reason why softwarization and modularization are being accepted and adopted by the mobile networking community is the availability of solutions implementing them, both commercial and open source. This availability of projects is caused by the relative maturity of these technologies, which are the enablers of the cloud computing success. In this sense, the mobile networking community is "lagging" with respect to computer science advances.

In this paper, we argue that, despite these recent achievements, we should "speed up" the adoption of newer developments from computer science. More specifically, we make the case for the adoption of "serverless" operation, a novel paradigm that appeared a few years ago supporting an extremely liquid approach to scalability and resource usage [3], [4]. With this approach, a tenant creates "on-demand" calls to specific functions that are then executed by an infrastructure provider. To support this, the software has to be decomposed down to its minimum building blocks (i.e., functions), in such a way that the code becomes platform- and server-independent, allowing unprecedented levels of scalability and resource efficiency. As further discussed in Section III, resource efficiency is of paramount importance in the multi-tenant scenarios envisioned in 5G.

To make the case for serverless mobile networking, in this paper we provide the following contributions:

- First, in Section II we describe the current landscape of network softwarization and modularization, comparing the advances in computer science and mobile networking.
- Then, in Section III we present the need for serverless computing as a key candidate technology for the next generation of Network Function Virtualization (NFV).
- Finally, we discuss the challenges introduced by this approach and propose possible solutions to address them in Section IV.

II. Evolution of Cloud and Mobile networking

There is a wide consensus among the research and industrial communities that future mobile networks will be software networks, due to flexibility and cost reasons (in fact, some functionality such as the Evolved Packet Core is already

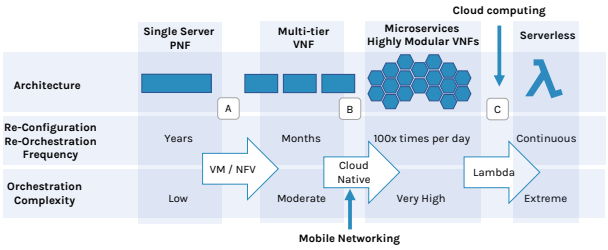

Fig. 1. Major transitions in the adoption of softwarization.

provided in specialized software running over general-purpose hardware). However, we still lack the ability to match the network demand at any point in time, i.e., a technology that is i) re-configurable over very fast periods, and ii) very granular, to reduce the cost of inaccuracies in the re-configurations in e.g. the access network [5].

A similar problem has already been tackled by the cloud computing community, which has continuously provided faster and more scalable solutions over the last decade. Additionally, these solutions have also made the system more flexible and open, enabling the appearance of new business models. In what follows, we first provide a quick overview of the evolution of cloud computing technologies and then review the current status of network softwarization.

A. Evolution of Cloud Computing

One major achievement in cloud computing took place in the early 2000s, with the appearance of new virtualization solutions such as Xen, VMWare or KVM. With these technologies, that efficiently exploited the novel virtualization extensions supported by the hardware, a new way of providing services "conquered" the cloud computing environment. It consisted of a more modular architecture that supported a higher re-configuration frequency but also required a higher management complexity. This achievement is marked with an 'A' in Fig. 1, where we illustrate the different transitions that we considered along three dimensions: architecture, re-configuration frequency, and complexity. For this first transition, the figure illustrates how the architecture evolved from monolithic functions to modular ones, supporting a change of operational timescales from years to months, but also increased complexity in the operation.

The second transition that we identify is marked with a 'B' in Fig. 1 and happened in the early 2010s. It was caused by the arrival of the so-called microservices paradigm [6], introduced by software architects to support a much finer granularity. This paradigm supports, for instance, that a database server can be split into many tailored microservices, each one fulfilling a specific functionality such as e.g., an account manager or the data storage system. This transition is driven by the availability of new virtualization technologies, such as Docker and LXC Containers, which allow the deployment and scaling of small virtual applications in a much more lightweight fashion, enabling also new coding practices such as DevOps [6].

Finally, the last transition that we can identify is the one marked with a 'C' in Fig. 1, namely, serverless architectures [3]. This recent paradigm, also known as Function as a Service (FaaS), is an extremely liquid approach to scalability and resource usage. With this approach, a tenant creates calls to functions, i.e., the minimum building block of a software component, which are served by the infrastructure provider. In this way, the software component becomes both platform- and server-independent, as the different functions of the same program could be served by different providers.

B. Evolution of Mobile Networking

There is currently a huge research effort on the softwarization of the mobile network. Among other efforts, 5G mobile communications are working towards the introduction of a fully softwarized architecture [7]. However, as compared to the cloud evolution, the telecommunications world is still halfway in this transition, despite the adoption of technologies such as Software Defined Networking (SDN) and Network Function Virtualization (NFV) which have helped towards the "softwarization" of network architectures, and the architectural trend towards their "modularization," with a clean separation between the control and the user planes. That is, the trend is to split, already at the architectural level, the formerly monolithic nodes into several smaller logical entities. Thus, the 5G Next Generation NodeBs (gNBs) can be split into centralized and distributed units, denoted as gNB-CU and gNB-DU, respectively [8], while core network components grow both in number and in functionality.

As depicted in Fig. 1, the telco world is lagging in the adoption of novel software paradigms. We are approx. at mark 'B' of cloud computing, with achievements such as:

- The standardization of 3GPP Release 15 [9], which specifies the Service Based Architecture (SBA). This represents a new paradigm for the 5G Core Network and is driven by the trend towards the modularization of the network. With this approach, the formerly static interface between different elements has evolved into a flexible bus, which hosts HTTP REST primitives between modules.
- The concept of Cloud-Native Network Functions (CNF), which is making its way into the current technology. In fact, there are already proposals for the design of cloud-native VNFs. However, they are in a very early stage and mostly involve Core Network VNFs only. We believe that the softwarization paradigm change shall involve all domains, including the most challenging one such as the RAN (as exemplified in Section IV).

Despite these achievements, we are still in the middle of this transition as the cloud-native paradigm has not been fully adopted into operational networks. This is caused by the poor agility of the current state-of-the-art solutions, and the fact that current VNFs are not truly agnostic to the underlying NFV infrastructure. While dynamic cloud resources orchestration algorithms are currently under study [10], the VNFs that will be running on such resources are still not optimized for this type of operation.

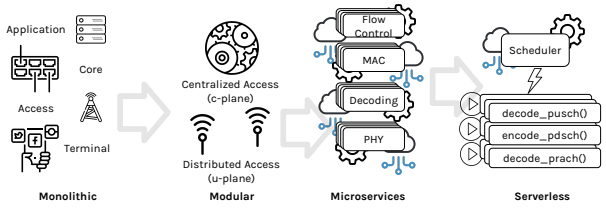

Fig. 2. Mobile network architecture evolution.

So even if the efforts towards the cloud-native transition of the NFV are still ongoing, the research community shall prepare for the next transition. This will introduce a complete re-design of the whole mobile protocol stack, which will certainly facilitate a dynamic resource orchestration and assignment, allowing hence higher efficiency. In this paper, we advocate for a mobile protocol stack that is (i) more efficient in terms of both resource and time granularity scalability, and (ii) capable to elastically adapt to the instantaneous demand. We claim that with such a protocol stack, the deployment and operational costs of the network will be reduced to their minimum. Given that this flexible and on-demand operation of the network closely resembles the current operation of cloud computing platforms, it should also follow similar principles, hence the name *Serverless Mobile Architectures.* We next formally introduce this concept and its potential advantages.

III. Serverless Mobile Architectures

In this section, we analyze the transition towards a serverless mobile network architecture. We first introduce the concept and then discuss the advantages, while we present the challenges to address along with some potential solutions next.

A. Concept

We start by describing what we mean by serverless mobile network architectures. To this aim, we illustrate in Fig. 2 the evolution of the different architectures to support a mobile service. Note that throughout this section, we use Radio Access Network (RAN) functions as examples, as they provide the most difficult scenario for serverless architecture given their tight execution constraints. The leftmost subfigure depicts the traditional **monolithic** paradigm (e.g., 4G networks), where functions are implemented in specialized pieces of equipment. In this case, software and hardware are tightly coupled, and it is not uncommon that different functions are indissolubility associated to the same piece of equipment, e.g., the Serving Gateway (S-GW) and Packet data network Gateway (P-GW).

The next subfigure illustrates a **modular** network architecture, represented by the Cloud-RAN (C-RAN) paradigm, where some control functionality traditionally associated with the antenna (i.e., the scheduling algorithm) is re-located to a central server. This change constitutes a shift from the monolithic approach, with some functions "freed" from their traditional association to monolithic pieces of hardware. These functions are now logically different pieces of software, whose execution can be placed in different parts of the network. As discussed before, this approach is also tackled by the architectural work, with the definition of standard interfaces among well-defined elements (e.g., the F1 interface between the gNB-CU and gNB-CUs).

The **microservices** architecture pushes the modular paradigm further, by decomposing the building blocks into sub-modules. Note that this is a logical division and that the actual implementation of the architecture needs to accommodate e.g. specific use-case requirements, this eventually resulting in fewer or more pieces of software. For the case of the RAN, this results in the protocol stack now being logically divided into physical layer processing, decoding, encoding, MAC, flow control, etc., each of them running in an independent execution environment and connected through synchronization APIs. This allows an easier scaling over a finer resource assignment strategy, which eventually leads to better resource utilization. Furthermore, some very recent proposals are pushing for microservice-based core network functions [11], showing that this increased modularity in the VNF design is catching momentum.

In this paper, we advocate for a **serverless** mobile architecture composed by atomic functions that can run independently on a cloud infrastructure. This independence contrasts with the tight coupling across functions in the other architectures, with strict timing considerations between modules. In a serverless approach, functions are dis-aggregated from the main scheduling logic and executed in the most appropriate server available. As Fig. 2 illustrates, for the case of User Plane Functions we envision, for instance, that the decoding of different Modulation and Coding Scheme could be made by different functions that could run in different executors, provided that some "loose synchronization" is guaranteed.

B. Advantages

Introducing the serverless operation as explained above brings several advantages to the network operation. Next, we build and extend the reasoning introduced in [3] to motivate the advantages of serverless mobile networking.

No server management: in the serverless paradigm, the functionality carried out by a VNF is broken into very fine execution environments (i.e., functions) that do not need to directly undergo into the classic lifecycle management (instantiation, run-time and decommissioning), but rather be scaled according to the real load and with a very fast pace in a "message broker" fashion. By moving this complexity to the network orchestration, this allows increasing the commodification of the network with a clear separation between the infrastructure and the services orchestrated therein.

No idling: operators usually provision the network based on the peak load. This is very inefficient at all network layers: at the access level, needless to say, but also at more centralized levels in which VMs or containers may be underused or even idling in trough loads. With the serverless paradigm, execution engines are spawned and operated just when and where they are needed. This is key for minimizing resource wastage in the network operation.

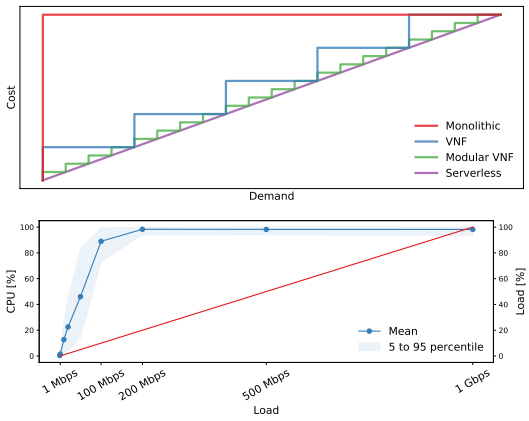

Fig. 3. The liquid scalability (top) and an empirical evaluation (bottom)

Liquid scalability: this is achieved by providing the highest *modularization* level. That is, specific functions of a VNF can be scaled according to the real demand, avoiding the scaling of the full VNF instead, and achieving the liquid scalability depicted in Figure 3 (top). We believe that this will be one of the fundamental pillars for the sustainable operation of the next-generation networks. In [5], we quantified the cost in terms of resource overhead of deploying and operating the infrastructure needed to support multi-service networks. This study showed that the efficiency (i.e., the number of resources used by a not multi-tenant network compared to a multi-tenant one) is very low, and just with a very dynamic network reconfiguration (such as the ones envisioned here) it is possible to improve these figures.

Figure 3 (bottom) depicts an empirical evaluation of the liquid scalability concept, by evaluating the CPU footprint of a state-of-the-art VNF (OpenVSwitch [12]), running inside a KVM Virtual Machine in Linux. This setup summarizes the "VNF" approach sketched in the upper part of Figure 3 as it can be scaled on a VM-basis (new VMs can be added or removed according to the load), this being the only way of scaling up or down according to the load. We can observe that this way of softwarizing clearly falls short when the objective is finer scalability. We can observe that while growing the offered load (in our experiments we span 4 orders of magnitude, from 100 Kbps to 1 Gbps) the resource footprint utilization has a clear "on-off" behavior. Indeed, the CPU utilization shortly hits close to 100 % utilization with offered loads that barely reach the 10% of the total. This means that there is still a lot of room for improvement in the software design of VNF with respect to IT resources consumption.

Pay-per-use network: although the pricing model behind the network slicing paradigm is not clear yet, it is to be expected that, at least for the software part, it will follow a classic approach in which tenants are charged on the number of CPUs, the amount of memory and bandwidth used. With a serverless approach, instead, tenants can be charged on a specific usage basis (i.e., number of times and duration of each function), allowing for a richer pricing model.

Customization: current mobile network technology provides only limited customization. For example, the currently envisioned resource models in 3GPP [13] target the Network Slice as a Service (NSaaS) paradigm, which is the telecommunication counterpart of the well-known Software as a Service (SaaS) paradigm employed in the cloud computing world. Under this model, service providers (or tenants) are allowed to select from an operator Network Slice portfolio one of the available templates (e.g., Enhanced Mobile Broadband). However, this provides limited customizability to tenants: the network provider still handles most of the management part.

IV. Challenges to Address

To achieve the above advantages, the serverless paradigm needs to provide: (i) new VNFs that allow for the wire speed execution VNFs while minimizing the number of resources needed for their operation, (ii) a new environment for the execution of such challenging VNFs, with minimal overhead, and (iii) a new Management and Orchestration framework that is capable to manage the rocketed complexity of the paradigm.

A. Challenge #1: New VNFs

The current way of implementing VNF is still very bound to the traditional way of implementing network functions. Current solutions do not embrace modularization: many commercial products are softwarized but very bounded to the hardware platform, while open source initiatives are practically mere translations of hardware functionality into software modules. To adopt the serverless approach, we need to change how VNFs are designed. To illustrate our point, in the previous section we focus on the user plane part of the RAN, but the same paradigm could be easily applied to other network functions. Our motivation is that, as the radio functions are the most resource-consuming ones (considering resources of all kinds: spectrum, transport network, and computational resources [14]), we expect that the transition towards high modularity will be especially beneficial for such functions.

We take as exemplary case study a well-known open source implementation of a 4G-LTE RAN stack, namely, srsLTE [15].We illustrate on the left part of Fig. 4 its threading architecture [16], which follows a *classical* layered architecture for a great extent and has remarkable modularity, in particular considering that the software has been designed for standalone operation. Still, the division is quite coarse, and there are additional issues that would prevent the use of a serverless approach, e.g., the physical layer does not distinguish between control and data channels, which are processed sequentially.

In this way, while the srsLTE design is perfectly valid for the working conditions initially considered by its developers, the architecture would benefit from a different design such as the one suggested in the right-hand part of Fig. 4. Here, the control and data plane processing are placed in different modules: while the control plane functionality on the L1 is "fixed" and has to be performed independently of the load, the user plane could be placed and executed in different threads (or even containers) to scale them according to the load. Alternatively,

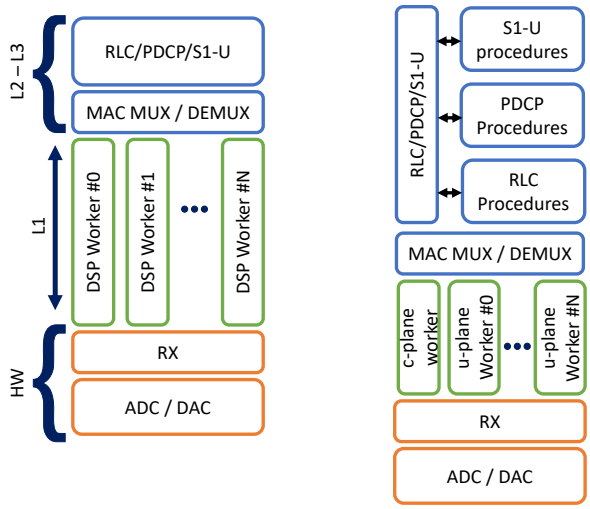

Fig. 4. The simplified threading architecture of the srsLTE software (left) and a possible serverless design of the software stack (right).

the software can be modified to support intelligent resource assignment schemes [17].

However, the current state of the art of networking (i.e., the Linux kernel) can hardly support this approach, as it is heavily API-based and requires fast inter-process communications. Moreover, current network functions (especially RAN functions) exhibit two fundamental characteristics that make them different when compared to other cloud computing applications: i) they impose very high load on the CPU (i.e., encoding and decoding of wireless frames) and ii) they have very stringent timing requirements as usually communication protocols are time-partitioned and need time synchronization. While there are CPU intensive services running in the cloud (e.g., Netflix video transcoding) these are not real-time. These timing constraints were barely a problem in the (now old) PNF paradigm, but it is a challenge to address since the arrival of novel technologies such as e.g. cloud C-RAN.

Making tasks aware of their execution time is usually not conceived as a problem in general cloud computing systems, which barely have to provide near to real-time outputs. The Linux kernel system provides tools to address this problem, but real-time software usually runs in rugged embedded devices for industrial purposes (e.g. robotics) or in dedicated data centers for fast pace trading in stock exchanges, not in the cloud. So, VNFs shall be re-designed to also take advantage of e.g., the real-time kernel primitives or used jointly with the elastic network function design as investigated in [14].

B. Challenge #2: Scalable interconnections

One key requirement for novel deployments, given the trends of mobile data consumption, is the ability to operate at wire speed. Being extremely fast on the data plane has been one of the main goals of research in wireless communications. However, the original virtualization platforms were not designed with this goal in mind. To address this issue,

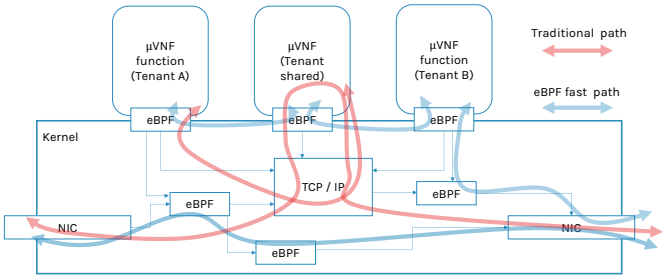

Fig. 5. eBPF-enhanced data path vs the traditional iptables-based approach.

the most common approach to achieve high performance has been *kernel bypassing*, through technologies such as DPDK and SR-IOV. These solutions skip the traditional Linux kernel networking stack based on `iptables` and enable a fast data path between the hardware and the application logic residing in the user-space. However, while this approach indeed increases the speed of the data processing, this hard link between the network card (i.e. the hardware layer) and the user-space reverts this approach to the traditional monolithic one, that leveraged on a tight hardware-software link. This hinders programmability as all the data traffic is offloaded from the kernel to the user-space, thus creating a vendor-specific link between hardware and software.

The above approach makes the management of the VNF very machine-dependent, so it is only valid for scenarios with a relatively small number of VNFs. Furthermore, it also has several drawbacks that make it difficult to integrate into a highly-dynamic scenario with a much larger number of software components (for each tenant, slice, and service, there might be multiple software functions). This would drastically increase the complexity of this kernel bypassing approaches, eventually resulting in two hardware infrastructure layers, namely, the kernel and the networking bypass, to manage the computing resources and the network platform, respectively.

We propose instead to integrate the data path back into the kernel. To this aim, i.e., to avoid the limitations from building on `iptables`, we propose to extend the Linux kernel networking stack with the adoption of enhanced Berkeley Packet Filters (eBPFs) [18]. These are pieces of code that can be dynamically injected into the kernel at run-time through a programmable interface. This approach allows managing the VNFs running on top of the kernel holistically, controlling all the aspects such as their CPU, memory, etc. in a unified way. eBPFs fit with a lightweight container-based approach, enabling strong security policies among them.

We illustrate our proposal in Fig. 5, where the bottleneck caused by the slow `iptables`-based design of the TCP-IP stack is removed thanks to the use of eBPFs. With these, a fast mesh of interconnected elements is created, providing an extremely scalable network infrastructure within the same host. Moreover, as BPFs are (i) executed in native code (e.g., x64), (ii) if needed, placed close to the NIC to ensure fast reaction times, and (iii) programmed with fast memory

mapping to the VNFs, the highest possible speed is achieved, comparable to DPDK according to recent studies [19].

C. Challenge #3: Precise orchestration algorithms

The serverless paradigm aims at the most efficient service provisioning, by accurately adjusting the resources deployed at any point in time to the actual demand. To benefit from this paradigm, it is essential to accurately estimate the demand required by a service and to forecast its envisioned resource consumption, to boost the multiplexing gains. To support this type of management, two main building blocks are required: (i) technical solutions to support flexible and fast resource re-orchestration at the finest granularity, and (ii) Big Data techniques that operate on historical data and anticipate future trends. The former should be achieved with the first two challenges (i.e., the use of functions instead of VMs and the integration of eBPFs into the kernel), while the latter requires the design of new techniques, as we discuss next.

We propose to use data-driven techniques to accurately characterize the future demand trends for a given service, this supporting a proactive and fine-grained orchestration of the network. This proactive management contrasts the current state of the art of resource orchestration, which is usually a reactive process based on load thresholds to trigger the scaling of VMs, and typical prediction algorithms that operate on very coarse and aggregated data. Moreover, the main goal of existing orchestration algorithms is usually fault-tolerance or self-healing, and not a higher efficiency. An efficiency-driven orchestration framework would provide an accurate forecast of demand, both in time and across resources (e.g., antennae, edge data centers, core clouds), to enable efficient provisioning of network services. Given the complexity of this type of operation, we believe that such orchestration is only possible if empowered with deep learning solutions [20].

The orchestration would work as follows. By starting from the historical demand of a given tenant/service, a deep learning architecture would provides the intelligent back-end for the Management and Orchestration of the network. By employing a deep learning technique, orchestration decisions are applied to the underlying NFV infrastructure that is used to host the VNF. This approach is currently being investigated by ETSI ENI [21] from the architectural point of view.

V. Summary

The cloud computing technology has already identified solutions for a more efficient service provisioning, through the microservices and the serverless paradigms, while the mobile networking community is lagging, implementing solutions based on Network Function Virtualization. In this paper, we advocate for the introduction of the serverless paradigm into the mobile network stack implementation. We believe the extreme flexibility of this approach is key to find the best trade-off between service customization and resource efficiency, but several research questions have to be solved before successfully introducing it: new VNFs shall be designed to exploit this paradigm, the underlying infrastructure needs to be prepared, and novel orchestration frameworks, possibly based on machine learning, are required.

Acknowledgements

This work was partially supported by the European Commission in the framework of the H2020 5G-PPP 5G-TOURS project and the 5G-EVE project.

References

[1] D. M. Gutierrez-Estevez et al., "The path towards resource elasticity for 5G network architecture," in *IEEE WCNCW*, April 2018, pp. 214–219.
[2] P. Rost et al., "Network Slicing to Enable Scalability and Flexibility in 5G Mobile Networks," *IEEE Communications Magazine*, vol. 55, no. 5, pp. 72–79, May 2017.
[3] P. Aditya, I. E. Akkus, A. Beck, R. Chen, V. Hilt, I. Rimac, K. Satzke, and M. Stein, "Will Serverless Computing Revolutionize NFV?" *Proceedings of the IEEE*, vol. 107, no. 4, pp. 667–678, April 2019.
[4] I. Sarrigiannis, K. Ramantas, E. Kartsakli, P. Mekikis, A. Antonopoulos, and C. Verikoukis, "Online VNF Lifecycle Management in a MEC-enabled 5G IoT Architecture," *IEEE Internet of Things Journal*, pp. 1–1, 2019.
[5] C. Marquez et al., "How Should I Slice My Network? A Multi-Service Empirical Evaluation of Resource Sharing Efficiency," in *Proceedings of 24th ACM MobiCom 2018*, 2018.
[6] A. Balalaie, A. Heydarnoori, and P. Jamshidi, "Microservices Architecture Enables DevOps: Migration to a Cloud-Native Architecture," *IEEE Software*, vol. 33, no. 3, pp. 42–52, May 2016.
[7] M. Condoluci and T. Mahmoodi, "Softwarization and virtualization in 5g mobile networks: Benefits, trends and challenges," *Computer Networks*, vol. 146, pp. 65–84, Dec. 2018.
[8] 3GPP, "NG-RAN; Architecture description," TS 38.401, v15.7.0, Mar. 2020.
[9] ——, "System architecture for the 5G System," TS 28.801, Mar. 2019.
[10] T. Taleb et al., "On Multi-Access Edge Computing: A Survey of the Emerging 5G Network Edge Cloud Architecture and Orchestration," *IEEE Communications Surveys Tutorials*, vol. 19, no. 3, pp. 1657–1681, Sep. 2017.
[11] V. Nagendra et al., "MMLite: A Scalable and Resource Efficient Control Plane for Next Generation Cellular Packet Core," in *Proceedings of the 2019 ACM Symposium on SDN Research*, ser. SOSR '19, 2019, p. 69–83.
[12] B. Pfaff et al., "The design and implementation of open vswitch," in *12th USENIX NSDI 2015*.
[13] 3GPP, "Management and orchestration; provisioning," TS 28.531, v15.2.0, Mar. 2019.
[14] D. Bega, A. Banchs, M. Gramaglia, X. Costa-Pérez, and P. Rost, "CARES: Computation-Aware Scheduling in Virtualized Radio Access Networks," *IEEE Transactions on Wireless Communications*, vol. 17, no. 12, pp. 7993–8006, Dec 2018.
[15] I. Gomez-Miguelez, A. Garcia-Saavedra, P. D. Sutton, P. Serrano, C. Cano, and D. J. Leith, "SrsLTE: An Open-Source Platform for LTE Evolution and Experimentation," in *ACM WiNTECH 2016*. New York, NY, USA: Association for Computing Machinery, 2016.
[16] J. A. Ayala-Romero *et al.*, "Demo: Vrain proof-of-concept – a deep learning approach for virtualized ran resource control," in *The 25th Annual International Conference on Mobile Computing and Networking*, ser. MobiCom '19, 2019.
[17] J. A. Ayala-Romero, A. Garcia-Saavedra, M. Gramaglia, X. Costa-Perez, A. Banchs, and J. J. Alcaraz, "VrAIn: A Deep Learning Approach Tailoring Computing and Radio Resources in Virtualized RANs," in *25th ACM MobiCom 2019*, 2019.
[18] Cilium, "BPF and XDP Reference Guide," 2020. [Online]. Available: https://cilium.readthedocs.io/en/latest/bpf/
[19] D. Scholz, D. Raumer, P. Emmerich, A. Kurtz, K. Lesiak, and G. Carle, "Performance Implications of Packet Filtering with Linux eBPF," in *2018 30th International Teletraffic Congress (ITC 30)*, vol. 01, Sep. 2018, pp. 209–217.
[20] D. Bega, M. Gramaglia, M. Fiore, A. Banchs, and X. Costa-Perez, "DeepCog: Cognitive Network Management in Sliced 5G Networks with Deep Learning," in *IEEE INFOCOM 2019*, April 2019, pp. 280–288.
[21] Y. Wang *et al.*, "Network Management and Orchestration Using Artificial Intelligence: Overview of ETSI ENI," *IEEE Communications Standards Magazine*, vol. 2, no. 4, pp. 58–65, December 2018.