NFV Orchestration Platform for 5G over On-the-fly Provisioned Infrastructure

Nazih Salhab, Rana Rahim, Rami Langar

NFV Orchestration Application for 5G over On-the-fly Provisioned Infrastructure

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

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

Rami LANGAR
LIGM-CNRS UMR 8049
U-PEM, F-77420
Marne-la-Vallée, Paris, France
rami.langar@u-pem.fr

***Abstract*—Multiple services with heterogeneous characteristics are anticipated with fifth generation of mobile communications (5G). Each requires an infrastructure tuned to serve requirements. To maintain affordability, operators will provide these services over a common infrastructure. We talk about provisioning Network Slices that are tailored to demands. A slice is a logical network built using Virtualized Network Functions (VNFs) that are enabled by leveraging Network Function Virtualization (NFV) and Software-Defined Networking (SDN). Orchestration is key for saving costs when tailoring infrastructures due to scarcity of resources. It enables operators to supply infrastructure with resources, meeting exactly with demands.**

In this demo, we present an on-the-fly provisioned cluster used to provision 5G Core Network (CN) as a Service (CNaaS). Our platform is designed for high-availability with self-healing feature. It uses Docker Swarm as orchestrator and Open Air Interface (OAI) for virtualizing the CN. We demonstrate resiliency, and the effortless on-demand scaling or orchestrated auto-scaling using bin-packing or spreading strategies for achieving elasticity.

***Index Terms*—Docker Swarm, Microservices, Orchestration, Auto-scaling, Elasticity, Cloud Architecture.**

I. Introduction

Fifth Generation (5G) of mobile network will serve different use cases with heterogeneous demands in terms of data rate, mobility, latency, reliability and energy efficiency. Reports forecast a continuation of the exponential increase of the number of smart-devices in addition to an anticipated 32 millions of Internet of Things (IoT) devices that will proliferate with 5G [1]. These devices will generate 44 Zettabytes of data per year [1]. Operators need to modernize their infrastructures to process this huge amount of data efficiently, with an adequate quality of experience at minimum spending to maximize their profitability. In this work, we are going to focus on the limiting factors on machines that are typically due to the finite and scarce nature of machine resources (Compute, Network and Storage) on infrastructure level. Resources provisioning is optimum when the "supply" meets exactly with the "demands". Prediction of required resource is used to prepare required resources ahead of time. However, on top of being computationally intensive and complex, predictions for scalability requires a sufficient amount of historical data to work properly, which is not always the case. In addition, sometimes, predictions are not even close to reality due to last minute changing circumstances or due to unforeseen factors. This results in losses due to over-provisioning. Elasticity in resources provisioning is a promising technology as it overcomes previous challenges on predictions' front. Network Function Virtualization (NFV) [2] is a network architecture that abstracts network nodes into building blocks as Virtualized Network Functions (VNFs) that will be deployed on a pool of resources enabling them to interconnect and chain easily to create communication services. Software Defined-Networking (SDN) [8] decouples the control plane from the switching plane making the network agile. Elasticity leverages NFV and SDN advents. It is well suited for cloud-based architecture using microservices [3]. On other hand, architects are changing the way they design their applications due to cloud emergence. Instead of monoliths, applications are decomposed into smaller and decentralized services so that they scale horizontally, by adding new instances as required. Leveraging cloud elasticity optimizes costs as it uses pay-as-you-go pricing [3]. Elastic resources are future-proof alternatives to predicted resources as they are reproducible and provisioned Just-in-Time. To maintain affordability, elastic resources have to be orchestrated, using automated processes, to provision and de-provision resources as needed. This fits well in a cloud architecture using microservices. Microservices are simple, indivisible tasks that are easily instantiated, replicated, and scaled in a short period [3]. A group of microservices forms a holistic communication service and is considered as a "Slice of a network". In this demo, we will demonstrate orchestration of 4G/5G Core Network (CN) using NFV and SDN in a Software-Defined Data-Center (SDDC) context [9]. Our contribution is an NFV Orchestration (NFVO) Application (APP) enabling to

- create a cluster to host microservices
- auto-scale services using different orchestration strategies.

Note that we made publicly available the source code of our NFVO-APP at [10]. The proposed NFVO-APP provides needed services while optimizing network resource utilization. The remainder of this paper is organized as follows. In section II, we describe our platform architecture, while in section III; we provide an overview of our planned demonstration.

II. Platform Description

We implement a virtualization of 4G LTE Evolved Packet Core (EPC) and 5G CN using "OpenAir Interface" (OAI-CN) [4] open-source software. We use Docker containers [5] to implement it as microservices. A container is a standard unit of software that packages up a code and all its dependencies so the application runs quickly, reliably on different computing environments and more additional security, as they are isolated in their contexts. Containers consumes fewer resources than Virtual Machines (VMs) as they share the same kernel with the host machine unlike VMs where the complete stack of a machine is implemented and thus consume additional memory. We orchestrate our system using Docker Swarm [5]. Swarm is a group of separate machines that will host our microservices. We implemented different strategies for provisioning microservices which are: "Bin-packing", and "Spread". "Bin-packing" assigns containers to one node until it is full before assigning them to another node. "Spread" assigns each container to the Swarm node with the most available resources, or in other terms, the least loaded node. The architecture of our platform is depicted in Fig. 1. It consists of provisioning several Swarm machines in order to secure High-Availability (HA). In our example, hierarchy is set to be one leader and five workers. For intercommunication, by default, we can rely on the internal routing mesh of the Docker Swarm manager to access a service on the target port of any node regardless of whether there is related microservice running on that node or not. Alternatively, we used an external load-balancer and reverse proxy for services with the internal routing mesh. Accordingly, HAProxy [6] or NGINX [7] is used to provide a floating IP address and caching service. In Fig. 1, service provisioning is done using "Spread" strategy to deploy LTE EPC implementation using OAI-CN [4]. This service is composed of several microservices. They are Mobility Management Entity (MME), Serving Gateway (SGw) and Packet Gateway (PGw) which are the building blocks of the EPC. On Swarm manager, we launch a script that runs indefinitely while monitoring sub-jacent services to control their size in term of Central Processing Unit (CPU) load as depicted in Fig. 2. It extracts CPU load on each service and according to predefined thresholds (threshold_l and threshold_h) takes a decision to scale-in/out (decrease/increase the number of microservices). A timeout phase follows any scaling in order to avoid snowball effect during execution.

III. Demo Overview

Our architecture complies with the five pillars of software quality, that are scalability, availability, resiliency, management, and security [3]. The demonstration aims to illustrate this compliance. Using one script, we start by creating an HA Infrastructure composed of a Docker manager with 5 workers topped by the load-balancer/reverse proxy and the visualizer [10]. Next, we provision a CNaaS. We show the effect of different scheduling strategies on the provisioning of CNaaS while changing on-demand the scale of the OAI-CN VNF. This will illustrate the availability, scalability and ease of management. We demonstrate self-healing capability to show the resiliency in two scenarios. First consists of retuning the service during run-time forcing it to be transported to another Docker-machine of the Swarm cluster. Second is by killing the microservice itself. Both scenarios end up by auto-recreation of the microservice. Third, we simulate an increase in load, so that we demonstrate the effortless auto-scaling using different orchestration strategies ("Binpacking" and "Spread") to show the auto-scaling of microservices without impacting other microservices due to isolation between Docker Swarm as security measures. 90 seconds demo video is available [11].

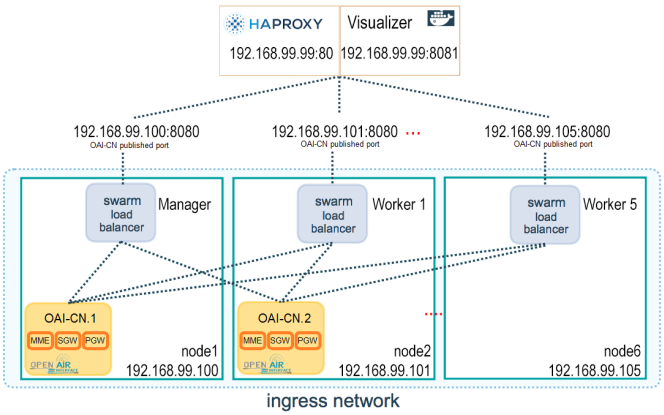

Fig. 1. OAI-CN (MME, SGW, PGW) provisioning with "Spread" Strategy

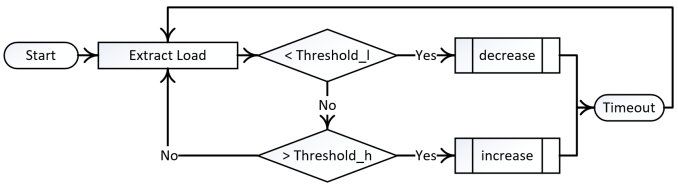

Fig. 2. Auto-Scaling Flowchart

Acknowledgement

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

References

[1] D Lund et al. "Worldwide and regional IoT-2020 forecast" AT&T, 2014.
[2] Bo Han et al. "Network Functions Virtualization: Challenges and Opportunities" IEEE Comm. Surveys and Tutorials, 2014.
[3] Microsoft, Cloud Application Architecture Guide, Online: docs.microsoft.com/azure/architecture/guide/.
[4] Eurecom, "OpenAirInterface", Online: openairinterface.org/, Jul. 2015
[5] DOCKER, Enterprise Container Platform, Online: docker.com.
[6] HAProxy, Reliable TCP/HTTP Load Balancer, Online: haproxy.org
[7] NGINX, Load Balancer & Reverse proxy, Online: nginx.org
[8] Open Networking Foundation, "TR-526 Applying SDN Architecture to 5G Slices", California, 2016.
[9] P. Marsh et al, 5G System Design - Architectural and Functional Considerations, Wiley, 2018
[10] Github, Online: github.com/nsalhab/docker-swarm
[11] Online: youtu.be/STZU_tV5ws8