Demo: Emulating 5G-Ready Mobile IoT Services

Moysis Symeonides*, Demetris Trihinas†, George Pallis*, Marios D. Dikaiakos*

* Department of Computer Science
University of Cyprus
{ msymeo03, gpallis, mdd }@cs.ucy.ac.cy

† Department of Computer Science
University of Nicosia
trihinas.d@unic.ac.cy

I. Introduction

The majority of IoT devices disseminate harvested data through the internet for analysis by cloud services. However, emerging applications, such as autonomous vehicle navigation, are impacted by the Round-Trip-Time between the IoTs and cloud. 5G networks and edge computing promise shorter RTTs by bringing compute and network resources closer to IoT. *Network slicing* is a key enabler for 5G networks, dividing a physical network among a variety of services under their individual needs. However, the design of a network slice impacts the performance of the mobile IoT applications with owners puzzled among the numerous slice configurations and options. For instance, the placement of network access points and available compute nodes, the wireless protocols, and the midhaul and backhaul QoS are crucial factors impacting service performance with the mobility of entities constituting this issue even more daunting.

Operators can address this challenge by purchasing a high-performance 5G network slice that provides radio units with numerous antenna elements and powerful compute nodes fully covering operational areas, however the latter is unrealistic due to the increased operational costs. Consequently, users have to select the minimum number of computing and network components that are capable of handling the volatile mobile workload and place them to maximize the network coverage. Thus, users need to perform multiple trials on diverse slices which is a rather time-consuming and costly procedure. In each trial, users have to design and lease the respective network slice, configure their physical IoT devices, deploy the IoT services, and monitor various KPIs.

To alleviate the difficulties in setting up real-world 5G testbeds, we will demonstrate 5G-Slicer [1], an emulation framework that facilitates the definition of mobile network slices through modeling abstractions for radio units, mobile nodes, trajectories, etc., while also offering realistic network QoS by dynamically altering -at runtime- signal strength. Moreover, 5G-Slicer provides an already realized scenario for a city-scale deployment that smart-city researchers can simply configure through a "ready-to-use" template, leaving 5G-Slicer responsible for translating it into an emulated environment.

II. 5G-Slicer Framework

Figure 1 provides a high-level overview of 5G-Slicer[1]. A typical deployment starts by either describing the application

[1] A detailed description of 5G-Slicer's modeling abstractions, implementation decisions and extensive experimentation can be found in [1]

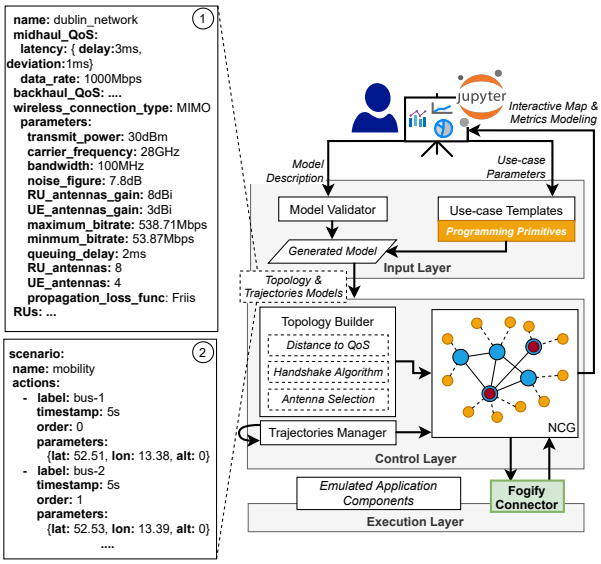

Fig. 1: 5G-Slicer overview

services and network fabric via the 5G-Slicer model specification or by parameterizing a "ready-to-use" testbed template. The **model specification** can denote a wide range of network slice parameters, including the position of compute nodes and RUs, network links and their QoS, mobile node traces, communication protocols and VNFs applicable on nodes. On the contrary , parametrizable **use-case templates** automatically produce 5G deployments for IoT applications. The output of each template is a programming view equivalent to a validated deployment description. The topology and trajectories are propagated to the control layer. Then, **Topology Builder (TB)** extracts from the description the network slice specification and any signal degradation models defined during the modeling process. With these, the TB produces a **Network Conceptual Graph (NCG)**, which contains the aforementioned information and will be used by the system for the runtime state propagation during the experimentation. The graph nodes represent network and compute devices annotated with information about their capabilities and deployed services, while edges denote the links between the nodes. The weight of each edge is determined by network QoS, incl. data rate, network delay, packet and error rate. Then 5G-Slicer translates the NCG to an emulated environment by utilizing the **Emulator Connector**. The connector is responsible for the emulation environment instantiation, and the deployment of the IoT application. 5G-Slicer's prototype offers a connector for the

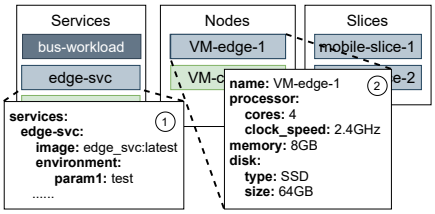

Fig. 2: Services & Compute Nodes

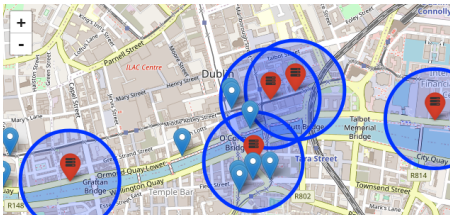

Fig. 3: 5G-Slicer Interactive Map

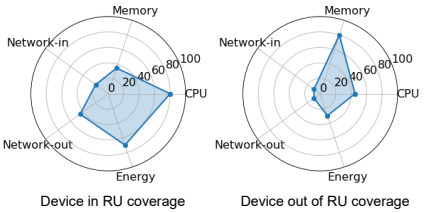

Fig. 4: 5G-Slicer profiling-kit (radar plot)

Fogify Framework, which is an interactive, multi-host, and scalable fog emulator [2]. However, 5G-Slicer is not bounded on Fogify, and users can extend the connector's interface to encapsulate other underlying emulators. Furthermore, the **Trajectory Manager** parses the traces and applies the updates on the NCG, and, in turn propagates these to the running emulated environment. One can view through an **interactive map** (Fig. 3) the traces of mobile nodes, their performance (i.e. cpu, energy), and the load imposed to MECs.

III. Demonstration

We will demonstrate the usability of 5G-Slicer by rapidly designing an emulation testbed, conducting repeatable experiments, and showing insights from a real-world IoT application.
Scenario. Let us suppose that a bus operator develops an IoT application that analyzes on-time streaming data from its fleet, and *would like to evaluate the service's response time on various 5G network slices to select the most efficient solution.* Towards that, the operator should communicate with local network providers, lease different 5G network slices in the operational area, configure each time the IoT devices, conduct multiple experiments, and monitor the deployments. However, taking advantage of the 5G-Slicer template, the operator simply introduces a smart bus network testbed. The template utilizes real-world mobility data from the Dublin smart city project[2]. In this, buses continuously report their locations and other attributes (i.e., environment conditions) to a central ITS to perform location-based analytic tasks (i.e., bus delay reporting), in collaboration with employed MECs that act as local data aggregators and are placed nearby bus stops.

```
1  from 5GSlicerSDK import 5GSlicerSDK
2  from experiment_repo import BusExperiment
3  5g_slicer_sdk = 5GSlicerSDK(
4                      'http://controller:5000',
5                      'docker-compose.yaml')
6  bus_exp = BusExperiment(
7      5g_slicer_sdk, num_of_RUs=5, num_of_edge=5,
8      num_of_buses=10, bounding_box=(...) )
9  5g_slicer_sdk = bus_exp.generate_experiment()
10 5g_slicer_sdk.generate_mobile_networks()
11 5g_slicer_sdk.deploy()
12 5g_slicer_sdk.generate_map('dublin_network')
13 5g_slicer_sdk.scenario_execution('mobility')
14 5g_slicer_sdk.undeploy()
```

Snippet 1: Smart Bus Testbed Scenario and Template

Snippet 1 depicts a brief example of how 5G-Slicer programming abstractions are used in the demo. Lines 1-2 import the 5G-Slicer SDK and the parameterizable smart bus testbed template. In Lines 3-5, the user introduces the Fogify Controller address (experiment orchestrator) and the docker-compose file, describing the available infrastructure and network resources along with the emulation configuration. An example of a mobile network configuration is shown in Fig. 1① while Fig. 2 ① and ② illustrate the definitions of services and compute nodes, respectively. Lines 6-8 configure the testbed according to user preferences, including the number of radio units, MECs, and buses, along with the operational bounding box. The *generate_experiment* method produces a new SDK object that captures a programming view of the 5G-Slicer model and materializes the testbed with the mobility scenarios. Line 11 deploys the testbed and Line 12 generates the interactive map, as presented in Fig. 3. With the *scenario_execution* method in Line 13, the user can run the mobility scenario that is generated from the datasets. Fig 1② depicts a excerpt from a bus mobility trace configuration. Finally, Line 14 finishes the emulation and releases all resources.

To assess different deployment settings, the demo will showcase the monitoring and analytics capabilities of 5G-Slicer that can be exploited through ipython notebooks for each experiment run. Our particular interest will be in showing how to optimize the placement of the MECs when their number, resource capabilities and network coverage change, so that the MEC nodes consume the majority of the mobile workload, rather than being routed to the, slower but more powerful, ITS cloud service. For instance, Fig. 4 depicts two utilization profiles of the IoT devices (bus data generators), namely when the device is in range of an RU and disseminates the IoT data, and when it is disconnected, storing data in-memory. The utilization profiles are illustrated via percentage-based (0-100%) radar charts, including CPU and memory usage, network traffic, and energy consumption.
Demo reproducibility.5G-Slicer (and all libraries used in this demo) are open-source and can run on a laptop. Thus, attendees can deploy both 5G-Slicer and the demo scenarios on their laptop, perform refinements to the experiments and see in real-time their impact on the running deployment.
Demo link. https://github.com/UCY-LINC-LAB/5G-Slicer-demo

Acknowledgement. This work is partially supported by the EU Commission through RAINBOW 871403 (ICT-15-2019-2020) project and by the Cyprus Research and Innovation Foundation through the COMPLEMENTARY/0916/0916/0171 project.

[2] https://data.smartdublin.ie

References

[1] M. Symeonides, D. Trihinas, G. Pallis, M. D. Dikaiakos, C. Psomas, and I. Krikidis, "5g-slicer: An emulator for mobile iot applications deployed over 5g network slices," in *IEEE/ACM IoTDI*, 2022.
[2] M. Symeonides, Z. Georgiou, D. Trihinas, G. Pallis, and M. D. Dikaiakos, "Fogify: A fog computing emulation framework," in *IEEE/ACM SEC*, 2020.