Deploying Smart City components for 5G network slicing

Bogdan Rusti, Horia Stefanescu, Marius Iordache, Jean Ghenta, Catalin Brezeanu, Cristian Patachia
Orange Romania
{bogdan.rusti, horia.stefanescu, marius.iordache, jean.ghenta, catalin.brezeanu, cristian.patachia}@orange.com

Abstract—The integration requirements between digital systems acting as enablers for vertical industries services and network layers represents a big challenge for multitenant vertical slice deployment initiatives over 5G infrastructure. Several concepts were developed, among them Smart City 5G-ready application based on cloud-native/microservice principles that proposes a separation between the orchestration of the 5G applications and the network services that support them. In this paper we are presenting the deployment phases of Smart City demonstrator focusing mainly on its first phase which covers the test bed deployment and the on-boarding process of application components. It is an advanced stage in the evolution of the project from the theoretical concept proposal [1], followed by the design and development phase as a collaboration between two 5G-PPP EU projects MATILDA and SliceNet [2] toward its completion as an automated solution for development, deployment and management of a Smart City 5G cloud-native application slice over the virtualized infrastructure. The target is to create and implement an end-to-end operational service framework starting from design and development to end to end orchestration over a 5G infrastructure through a One-Stop API assuring the end-to-end management, control and orchestration of the slice.

Keywords—5G, Marketplace, 5G-ready application, Cloud-native, Network Slice, Slice Intent, IoT, Sensor, Smart City, Matilda, SliceNet

I. INTRODUCTION

The connectivity and computing capacity that 5G enables together with Smart City cloud-native application deployment will create the premises and the technological capabilities to perform the end to end deployment and in life management of a smart lighting solution towards network virtualized infrastructure in an automated way without requiring any human intervention from the infrastructure provider. Integrating these capabilities on a Smart lighting solution assures individual remote management of streetlight lamps and autonomous operation based on predefined schedules and light sensors input.

Three main layers are embedded in 5G Smart City application architecture: (1) the Development Environment and Marketplace [3] to support all pre-deployment steps of a 5G-enabled application; (2) the 5G-ready Application Orchestrator (VAO) [4] layer supporting in-life slice intent and adaptation of the application to its service requirements by using the optimisation schemes and intelligent algorithms to provide the needed resources across the programmable infrastructure; (3) the Network virtualized infrastructure layer responsible for set-up and management, deployment and operation of vertical application.

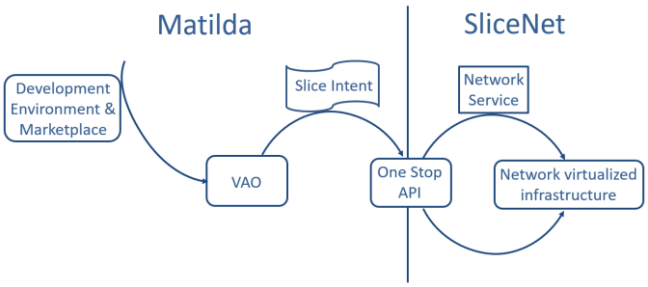

Fig. 1. Smart City application layers

As depicted in Fig. 1 the logical flow among the specified layers, the communication between them as explained later in the documents is enabled using a set of metamodels [5]. The communication between the first two layers developed in MATILDA [6] with the third one is performed through a One-Stop API [7] solution both addressed in SliceNet [8] as a joint effort between the two projects.

In the next section the Smart City 5G demo architecture developed to materialize the envisaged architecture is presented. In section III the demonstrator setup and the application deployment flow are exposed. The performance evaluation metrics output is analysed in chapter IV. The main conclusions of the demo are summarized in section V.

II. SMART CITY DEMO ARCHITECTURE

ORO test bed solution architecture will be developed for testing and validation of the Smart City use case and it will be deployed as a small-scale network, which will combine layers from both projects with the aim to create a logical infrastructure with appropriate isolation and on de-demand dedicated resources adapted to the application needs.

The Smart City use case demonstrator has in composition the following application components: (1) component 1, an IoT Aggregator to pass the raw data information from device sensors to the IoT platform; (2) component 2 the IoT Platform (Thingsboard.io) with the

978-1-7281-0546-8/19/$31.00 ©2019 IEEE

role of provisioning, processing, visualization and device management role; (3) component 3 is configured as a Dashboard and administration component; (4) component 4 - Monitoring application; (5) component 5 - Ticketing system component; (6) component 6 - Billing application component;

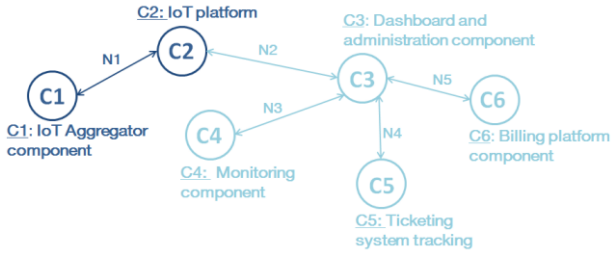

Fig. 2. Smart City Application Graph

For the proper placement of components into virtualized environment and to follow cloud specific application requirements an integrated Development Environment was created to design the Smart City lighting solution as a distributed application. The application is a composition of several chainable cloud native components stored in a Marketplace repository to be logically linked in the form of an application graph descriptor, containing the configuration options and operational policies to describe how the application components should adapt their execution mode during runtime. The information is then further used by the Vertical Application Orchestrator to enable the creation of an appropriate application-aware network slice over Network virtualized infrastructure. The slice deployment request is sent through One-Stop API application (OSA), with the role of translating the slice intent associated to Smart City application graph, into a valid allocation demand.

The demonstrator deployment is planned in three phases:

- Phase 1 – completed and covers the deployment of the test bed and process of development and validation for the first two components of the demonstrator (C1, C2). Additionally, the dockerizing process of component C2 and the on-boarding in the Marketplace is planned. These steps are not applicable for component C1 (fixed component). At this stage the upper layers developed in Matilda are not linked with the virtual infrastructure of the test bed addressed in SliceNet.
- Phase 2 - is under development addressing the integration of ORO test bed network virtualized infrastructure developed in SliceNet with Matilda upper layers, through OSA solution. Also, the development, dockerization and on-boarding of the rest of components (C3-C6) is planned. In this phase the slice deployment is not operating automatically, a static APN being provisioned for the setup.
- Phase 3 - in this phase the in-life management process is envisaged addressing the automation of virtual infrastructure resource scaling (VM's,) and radio resource scheduling.

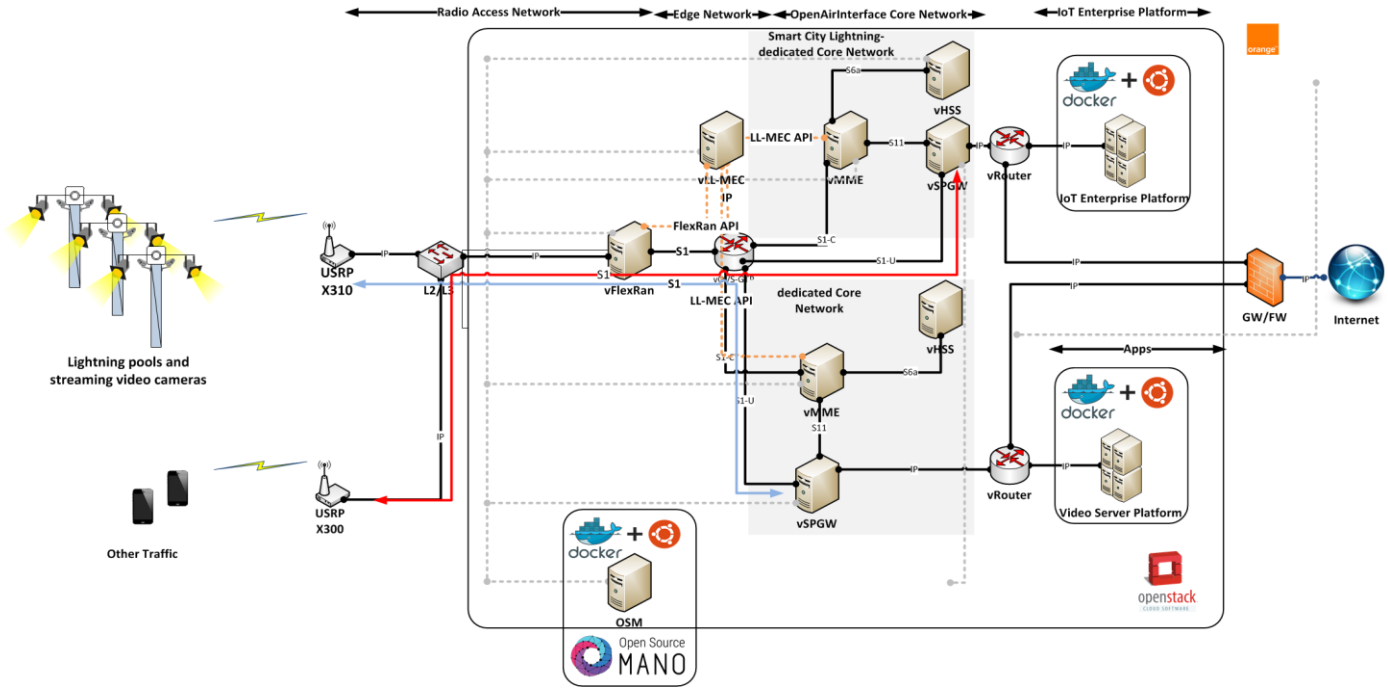

Fig. 3. Smart City testbed infrastructure

The test bed infrastructure presented in Fig. 3 designed and deployed in this projects, is composed by a suite of hardware and software components containing:

- physical network elements: Cisco NX-OS 9k switches, ASR9k routers and Fortinet 1500D firewall

- physical hardware servers and storage, used as the support environment for the virtualized implementations. All virtualized network elements are deployed over two compute clusters, hosted on a dedicated HPE servers: (1) ProLiant DL360 Gen10(1) composed by: 2 processors, 12 core/processors, 512GB RAM, 2.4 TB HD, Network adapter 1Gbps/2ports, Ubuntu 16.04 Server LTS; (2) HP Servers - ProLiant DL380 Gen9(3) composed by: 2 processors, 12 core/processors, 128GB RAM, 600/1.2T GB HDD, Network adapter 1Gbps/6ports, Ubuntu 16.04 Server LTS
- Smart Lighting PNF (IoT connector and Gateway), as the physical component used to connect the specific use application encapsulated traffic to the IoT platform and applications
- radio remote baseband units
- test bed lamps (Fig. 4) - connected to the IoT aggregator through LoRA WAN controller

Fig. 4. Smart City lighting lamps

The software components refer to:

- components for testbed virtualization instantiation - OpenStack framework
- software components for NFV/VNF instantiation
- software components for Orchestration - Open Source MANO and the on boarded VNFs templates

The first step of the setup is the testbed deployment covering the entire process of component integration at both infrastructure and service level. A prerequisite for this phase is the integration and the configuration of the hardware and software components. Four physical servers and OpenStack framework (containing: Ceilometer, Cinder, Glance, Gnocchi, Heat, Keystone, Murano Netdata, Neutron, Nova, Openvswitch-agent, RabbitMQ, KVM services) were considered for deployment of NFV/VNF ETSI MANO [9] virtualized infrastructure. A software component for Docker [10] containers and Kubernetes [11] as the container orchestrator is also supported.

The infrastructure layer was design by using the OpenStack Ocata composed by a set of software tools for building and management of the virtual computing, networking and storage resources of the test bed. The software tools are distributed on four physical machines: (1) the controller server, where are hosted all the OpenStack service databases and control software components, acting as a Virtual Infrastructure Manager (VIM); (2) the management and orchestration server where are hosted all the management and orchestration software service components; (3) two compute cluster servers hosting and enabling the virtualization layer of the physical resources.

The service layer deployed on the computing clusters of the test bed was created by integrating Open-Air Interface (OAI) [12] solution with the following components:

- radio component: universal software radio peripheral (USRP), LTE/LTE-M capable
- Mosaic5G [13] OAI-FlexRAN used for mobile network functions virtualization from the access network to the evolved packet core (EPC).
- Mosaic5G OpenVSwitch GPRS Tunneling Protocol, enabled to control and transport all the GTP encapsulated traffic
- Mosaic5G OAI-SPGW, OAI-MME, OAI-HSS
- LL-MEC [14] platform is a SDN using OpenFlow that is facilitating the communication between LL-MEC and underlying infrastructure

III. SMART CITY APPLICATION DEPLOYMENT FLOW

The application deployment flow addressed in phase1 of deployment, starts with the on-boarding process of Smart City application component in the Development Environment and Marketplace (Fig. 5). The process flow is performed through a REACT.JS [15] user interface, specially created for this scope.

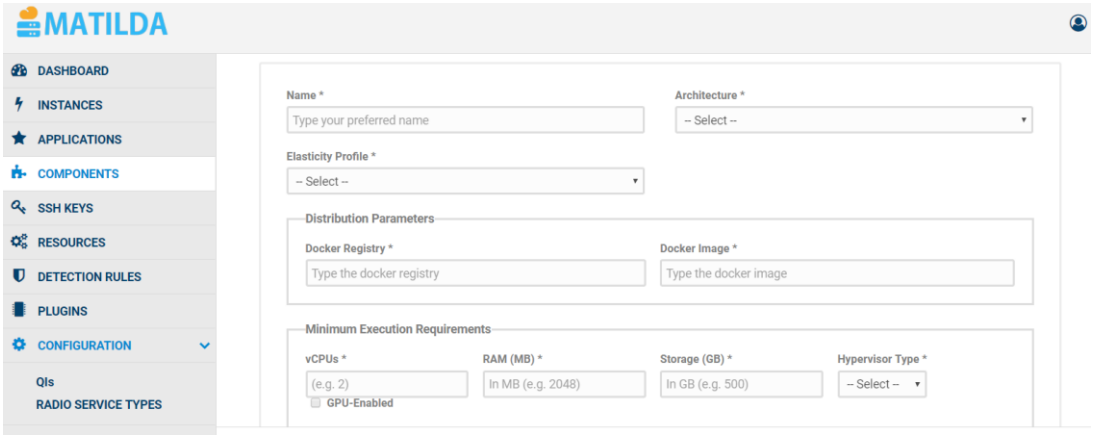

Fig. 5. Marketplace interface

Smart City application components and interconnecting links are developed based on predefined metamodels requirements. For on-boarding preparation, each application component is translated first into a multi-container application using Docker compose tool (Fig. 6).

```
DEV [slicenet@CentOS72x86-apps docker_iot]$ ls -la
total 76
drwxr-xr-x   5 root root  4096 Feb 20 17:05 .
drwxr-xr-x. 11 root root  4096 Feb 20 14:48 ..
-rw-r--r--   1 root root  1095 Feb 19 13:30 docker-compose-db.yml
-rw-r--r--   1 root root  3929 Feb 19 18:43 docker-compose-thingsboard.yml
-rw-r--r--   1 root root   482 Feb 18 18:45 .env
drwxr-xr-x   5 root root  4096 Feb 19 18:30 haproxy
-rw-r--r--   1 root root 12224 Feb 19 22:25 http_docker_file
-rw-r--r--   1 root root   841 Feb 18 18:19 kafka.env
-rw-r--r--   1 root root    97 Feb 18 18:21 tb-coap-transport.env
-rw-r--r--   1 root root   105 Feb 18 18:21 tb-http-transport.env
-rw-r--r--   1 root root   205 Feb 18 18:21 tb-js-executor.env
-rw-r--r--   1 root root    97 Feb 18 18:22 tb-mqtt-transport.env
drwxr-xr-x   5 root root  4096 Feb 19 14:45 tb-node
-rw-r--r--   1 root root   267 Feb 18 18:22 tb-node.env
-rw-r--r--   1 root root   369 Feb 18 18:23 tb-node.postgres.env
drwxr-xr-x   5 root root  4096 Feb 18 22:03 tb-transports
-rw-r--r--   1 root root   161 Feb 18 18:24 tb-web-ui.env
```

Fig. 6. Aplication components containerization

The application components and associated links are logically bound together in the form of an application graph composed by six components:

- Component 1 (C1) is an IoT Aggregator which sends in a secure manner the raw data information from device sensors to the IoT platform. It integrates a two-way communication between IoT platform apps and the devices sensors, assuring the interoperability, scalability and simplified communication.
- Component 2 (C2) - IoT Platform (Thingsboard.io) with the role of provisioning, processing, visualization and device management (Fig. 7)

Fig. 7. Component C2

- Component 3 (C3) will be configured as a Dashboard and administration component for managing lighting
- Component 4 (C4) - monitoring application allowing the user to monitor activity of the service.
- Component 5 (C5) - ticketing application component it helps the final user to open/close/manage all tickets regarding Smart Lighting service.
- Component 6 (C6) - billing application component for C2 data usage

The set of deployment and operational requirements defined for each application component and graph links are further used for creation of the slice intent flow. Each application component requirements will be exposed in a XML slice intent descriptor file format generated through XSD to be further used during instantiation phase. The requirements are defined based on metamodels that includes a set of fundamental complexType [5] elements that uniquely describe each component in the entire service graph. Requirements related to CPU, RAM and storage capacities along with constraints related to the hosting node of each component are declared for each component. Additionally, specifications regarding the virtual links between components (maximum acceptable end-to-end delay, maximum acceptable packet loss rate) are defined.

Component C2 on-boarding specifications:

ComponentIdentifier: IoTPlatform
Distribution:
Image Descriptor: IoTPlatform
Repository Descriptor: URL 2
ExposedInterface:
InterfaceIdentifier: C2C1 interface
InterfaceType: ACCESS
TransmissionProtocol: TCP/UDP
InterfaceIdentifier: C2C3 interface
InterfaceType: CORE
TransmissionProtocol: TCP/UDP
RequiredInterface:
GraphLinkIdentifier: C2C1 connection
ComponentIdentifier: IoT Aggregator
InterfaceIdentifier: N1 interface
GraphLinkIdentifier: C2C3 connection
ComponentIdentifier: Dashboard

InterfaceIdentifier: N2 interface
Configuration:
MinimumExecutionRequirements:
VCPUs: 12
RAM: 16384 MB
Storage: 61440 MB

IV. NEXT STEPS

The flow, that will be developed in last two phases, continues with the instantiation procedure, where the slice intent descriptor is sent to OSA application to be translated into proper requests for deployment over virtualized infrastructure. The instantiation requests will be processed by the management plane of the test bed virtual infrastructure and the end-to-end slice instance is created. The in-life management of the slices designed to assure an automated scaling of virtual infrastructure processing capacities and radio resources will assure the end to end QoS defined during slice intent creation.

After the implementation of Smart City demonstrator, a set of KPI's and acceptance criteria measurements will be implemented and evaluated based on the counters information, alarms events from both the physical and virtual network resources and smart lighting deployed slice. Two set of KPI's are foreseen in this moment, one for the service performance evaluation and the second one for the Smart City demo platform. The proposed Smart City service KPI's were specifically designed to provide an end view about the performance during different deployment scenarios. To see for examples how the proposed infrastructure is scaling when concurrent slices are running in parallel and what is the impact bring to the Smart City quality of service. The list would be enriched with additional metrics in step with project deployment evolution (see the tables below).

TABLE I. SMART CITY SERVICE KPI'S

KPI	Description	Threshold
Device status	Evaluates the number of smart light sensors deployed on test bed platform.	100 Smart Light sensors
Service Availability	Calculated as service down time/total time, reflects in percentage the availability/stability performance of Smart City service	>99.99%
Device bandwidth capacity	Evaluates the transfer capacity volume of information collected from sensors to IoT platform.	~0.1 Mbps
Total slice bandwidth	Evaluates the transfer capacity volume of aggregated information from sensors to IoT platform. Calculated as devices number x bandwidth/device (helpful for VNFs system parametrization)	~ 100Mbps
End-to-end Latency	Measures packet round trip time from IoT platform to device sensor.	< 300 msec
Jitter	Evaluates packet delay variation in latency between IoT platform and device sensor.	~100 msec
Packet Loss	Shows the percentage of packets lost during transfer between sensors and IoT platform. The Smart Lighting service is not critical therefore retransmission is being allowed, without affecting end-to-end application functionality.	< 0.1%

TABLE II. SMART CITY DEMO PLATFORM METRICS

KPI	Description	Threshold
Availability	Calculated as network down time/total time, reflects in percentage the availability/stability performance of Smart City demo platform	> 99.99%

V. CONCLUSIONS

In this paper we have exposed the Smart City demonstrator architecture together with the description of its main deployment flow phases. All six components of the solution were described starting with capabilities requirements definition, cloud-native transformation through dockerizing process for on-boarding in the Marketplace repository, description of application graph requirements based on slice intent predefined metamodel and implementation on test bed virtual infrastructure managed by a dedicated interface between the two projects layers. An important research effort was involved to identify and configure the open-source software tools used for the entire management of the virtual computing, networking and storage resources of the test bed.

In-depth description of the test bed infrastructure was presented highlighting the development evolution steps. Also, the service test bed layer composition containing a set of VNF's (vRAN, vHSS, vMME, vSPGW) chained within VNF-FG was emphasized.

The first deployment phase of Smart City demonstrator covering component C2 on-boarding and test bed infrastructure deployment is completed. The next steps, already on-going are targeting: (1) the on-boarding of the rest of application components; (2) the creation of the mechanism for automated Smart City slice instantiation by deploying of One-Stop API solution; (3) in-life management flow design based on defined acceptance criteria and thresholds.

ACKNOWLEDGEMENT

This work has received funding from the European Union's Horizon 2020 research and innovation program under grant agreement No 761898 project Matilda and grant agreement No. 761913 project SliceNet.

REFERENCES

[1] Bogdan Rusti; Horia Stefanescu; Jean Ghenta; Cristian Patachia. Smart City as a 5G Ready Application. 2018 International Conference on Communications (COMM). 14-16 June 2018. DOI: 10.1109/ICComm.2018.8484752/

[2] B. Rusti, H. Stefanescu, M. Iordache, J. Ghenta, C. Patachia, P. Gouvas, A. Zafeiropoulos, E. Fotopoulou, Q. Wang, J. A. Calero, "5G Smart City Vertical Slice," [online] Available: https://pdfs.semanticscholar.org/5dcd/9366fea491ed57aca6892cb73be69621a2ad.pdf

[3] D1.1 - MATILDA Framework and Reference Architecture [online] Available: http://www.matilda-5g.eu/index.php/outcomes

[4] D4.1 - MATILDA Framework and Reference Architecture [online] Available: http://www.matilda-5g.eu/index.php/outcomes

[5] D1.2 - Chainable Application Component & 5G-ready Application Graph Metamodel. [online] Available: http://www.matilda-5g.eu/index.php/outcomes

[6] 5G Infrastructure Public Private Partnership (5G PPP) - Matilda [online] Available: https://5g-ppp.eu/matilda/

[7] D2.2 – SliceNet Overall Architecture and Interfaces Definition [online] Available: https://slicenet.eu/deliverables/

[8] 5G Infrastructure Public Private Partnership (5G PPP) - SliceNet [online] Available: https://5g-ppp.eu/slicenet/

[9] Open Source MANO [online] Available: https://osm.etsi.org/

[10] Docker container [online] Available: https://www.docker.com/resources/what-container

[11] Container Orchestration [online] Available: https://kubernetes.io/

[12] OpenAirInterface, Jul.2018, [online] Available: http://www.openairinterface.org

[13] Mosaic-5G.io, Jul. 2018, [online]. Available: http://mosaic-5g.io/

[14] Mosaic-5G LL-MEC, Nov.2018, [online]. Available: http://mosaic-5g.io/ll-mec/

[15] Building User Interfaces [online] Available: https://reactjs.org/