# 5G AutoMEC – Boosting edge-to-edge service continuity for CAM in a sliced network

Girma M. Yilma*, Umberto Fattore*, Marco Liebsch*, Nina Slamnik-Kriještorac[†], Andreas Heider-Aviet[‡], and Johann M. Marquez-Barja[†]
* NEC Laboratories Europe GmbH, Germany.
[†] University of Antwerp - imec, IDLab - Faculty of Applied Engineering, Belgium.
[‡] Deutsche Telekom AG
E-mail: {Name.Surname}@{neclab.eu | uantwerpen.be | t-systems.com}

***Abstract*—Network function virtualization and edge computing in mobile networks are key enablers in the evolution of recent standards for a 5$^{th}$ Generation (5G) mobile communication system towards a comprehensive 5G ecosystem, which enables the deployment of customized networks and service access for various industry verticals by means of clearly defined and deployed network slices. In particular, the automotive industry can leverage low-latency access to services hosted along the distributed network edge, in support of a large variety of use cases. Network slices typically implement the requested service and network according to a set of defined requirements, as well as performance and latency bounds, while using a defined resources budget. With connected cars following different mobility patterns, the automotive sector represents a pretty agile customer of such network slices. This makes the management of network and edge computing resources a key challenge to tackle in order to balance the resource utilization, and the previously agreed and expected service levels. In this paper, we analyze the benefit of smart mobile edges and the use of machine learning to anticipate resource demand at distributed mobile edges in an automotive scenario. Finally, we experimentally show the feasibility to treat $\mu$-slice resources efficiently by using an OSS-based prototype for the orchestrated edges, which is currently being developed for an automotive trial in Europe.**

***Index Terms*—5G, CAM, Cloud-native Edge, Network Slices, Resources, Service Continuity**

## I. Introduction

Service- and Network Function Virtualization (NFV), as well as the Management and Orchestration (MANO) systems, play a vital role in the provisioning of a 5G Ecosystem. Such ecosystem enables the customization of services as well as access to them by providing a suitable topological placement in decentralized cloud resources, including an on-demand function re-configuration (e.g., instantiation, scale, and migration). The inter-play between a 5G cellular system with NFV and programmable networks enables a quite powerful and flexible ecosystem for tailored service provisioning as well as runtime re-configurations. Adjustments may be required at the used infrastructure and the deployed functions, aiming at the best match between the utilized resources (e.g., cloud compute and storage, network infrastructure and wireless) and the quality of service experienced by clients (e.g., dependent on throughput, latency, availability, etc.).

The principles of Network Slicing (NS) have been thoroughly investigated in various projects, and in the meantime they have been adopted by standards, such as 3GPP [1]. In particular, NS contributes to the provisioning of customized networks and services to tenants (e.g., vertical industry) by: i) defining requirements associated with a requested service, ii) providing an isolated treatment of such definitions, iii) implementing the associated service- and network functions, and iv) using the required resources and enforcement of suitable policies.

Furthermore, besides NFV and NS, the third key enabler of a suitable ecosystem for customized provisioning of networked services is driven by the ETSI ISG MEC for Multi-Access Edge Computing (MEC), which represents a platform for the provisioning of services in distributed cloud resources that are topologically closer to (mobile) clients [2]. This helps to localize functions and data plane traffic, thereby enabling potentially short routing paths between clients and a service instance on one of the MEC platforms. To benefit from MEC deployments in a highly mobile automotive scenario, the MEC platform, data plane routing policies, and infrastructure resources, must be aligned to support the situation of connected clients on the road. For example, experiencing an unexpected increase in mobile clients in a certain location may overload a service on a single edge resource, which may result in service quality degradation and violation of agreed and expected service levels. In this paper, we show how an agile resource management of distributed edge resources in an NS-enabled ecosystem can be accomplished without exceeding the overall resource budget assigned to that slice. Furthermore, the applied machine learning helps to anticipate the upcoming resource demand, and to give the required time to the system to scale and re-configure the edge resources in advance. The proof of edge slicing and resource management per $\mu$-slice resources assigned to a single edge resource is experimentally shown in an orchestrated container-virtualization-based edge platform, which is being developed for an automotive trial in the EU's 5G-CARMEN project.

## II. Orchestrated Edges – System View

So far, various projects have investigated on the orchestration of a network in an end-to-end context, i.e., from a central

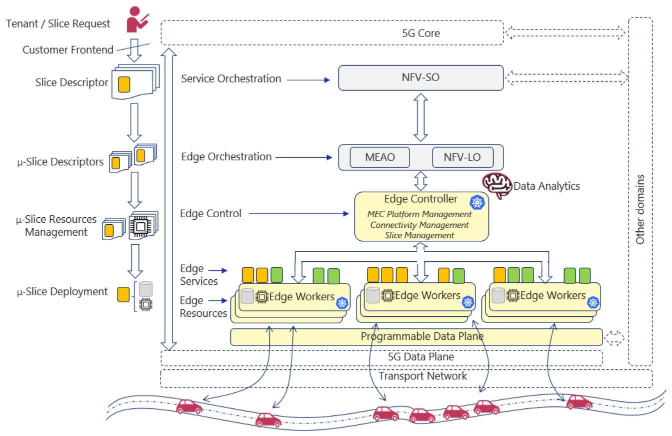

Figure 1. Deployment of service slices in the distributed container edges.

office to a mobile client, thereby managing the functions, resources, and policies, which are deployed in the core, transport, and the radio access networks. In this paper, we focus on the orchestration of distributed edge resources to support the deployment and continuity of services for Connected and Automated Mobility (CAM). Thus, Fig. 1 sketches a high-level view of the specified functional architecture [3] and comprises an NFV Service Orchestrator (SO), which can select orchestrated edge resources according to a deployment strategy, and enforce the instantiation of an image of the requested CAM service at the selected edge through an NFV Local Orchestrator (NFV-LO). For a large-scale deployment, the NFV-SO can request multiple edge resources to deploy such service.

In particular, the NFV-LO and the Mobile Edge Application Orchestrator (MEAO) [2] treat the service deployment per its description in a Network Slice Template, and admit the enforcement of the service through an Edge Controller function. The Edge Controller handles multiple distributed servers in one or multiple edge clouds, which are denoted as worker nodes that provide the local hardware resources for the deployment of service instances. The federation interfaces are sketched at the NFV-SO and the NFV-LO level to accomplish roaming and cross-border scenarios. The additional roles of the Edge Controller include i) slice management, ii) connectivity management, iii) network programming for traffic steering, as well as iv) interfacing with the 5G Core network for receiving client mobility related event notifications, which may require re-configuration of services and traffic steering policies within or between local edge clouds. In case more mobile clients access edge services from a certain location, the orchestration system and the Edge Controller need to provide and re-configure the associated local edge resources accordingly. To provide a mobile access to the topologically closest edge service, and to distribute the load properly between all edge resources utilized for a service, the transfer of a client's session state might be needed, from a service instance on one edge to an instance on another edge.

In case the deployment of distributed service instances is constrained by the system's slice description and an associated resource limit, the orchestration system and the Edge controller need to distribute the allowed resource budget over all edge resources, where the service instances need to be deployed. Segmentation of a Network Slice into Sub-Network Slices (e.g., for the edge network) is described in standards [1], whereas we introduce the notion of a *$\mu$-slice resource*, which is a portion of the resources assigned to an edge Sub-Network Slice. In the following section, we describe how the distributed edge resource management can be accomplished by partitioning the available slice resource budget over multiple edge worker nodes according to resource demand. We briefly show how the chosen software principles for the development of a container-virtualization based orchestrated edge can be used to enforce $\mu$-slice resources on an edge worker node to take additional load from connected cars, which is anticipated by machine learning principles applied to the mobile edges.

## III. Slicing-enabled Cloud-native Edges

Containers such as Docker[1] are gaining momentum as a way to develop network applications and services in a modern telco world, due to their small footprint compared to virtual machines, and their seamless/flexible integration with modern cloud-native micro-service architectures, including continuous integration, development, and deployment strategies. Whereas containers are fast to start, stop and restart, virtual machines have better level of isolation.

The 5G Service-based Architecture (SBA), which is the main driver for 5G control plane, benefits a lot from container based applications/services as containers have proven in cloud data centers for this kind of architecture. Hence, container network functions (CNFs) are becoming the standard *defacto* for MEC/Edge cloud network services/applications [4].

Though containers are good in managing and packaging application specific dependencies (package once and run anywhere), container orchestration solutions such as Kubernetes[2] are required for automating deployment, and management at a scale (e.g in a production environment). Besides Kubernetes, many other solutions such as Apache MESOS[3], Nomad[4], Amazon Elastic Container Service[5], etc., are joining the cloud native computing foundation[6]. In our work we rely on Kubernetes for the following key reasons: I) It is open-source with active community support II) Maturity and wide acceptance both in academy and industry.

A Kubernetes cluster is a set of worker node machines for running containerized applications. A cluster contains a master node, which controls the worker nodes assigned to the cluster and maintains the desired state of the cluster, e.g. which applications are running, which container images they use and which worker nodes to run applications and workloads.

[1]Docker: https://www.docker.com/
[2]Kubernetes: kubernetes.io,
[3]Apache MESOS: mesos.apache.org
[4]Nomad project: nomadproject.io
[5]Amazon ECS: https://aws.amazon.com/ecs/
[6]The cloud native computing foundation: https://landscape.cncf.io/

### A. Main Kubernetes Objects for Slicing

**Pods** are the smallest (i.e., atomic) units of Kubernetes objects, and they contain one or more containers. The containers in a single Pod share the same network namespace, and can connect to each other via the loop-back interface. Furthermore, Pods are stateless and local to the specific Kubernetes cluster. **Service** objects are used to expose applications running in Pods for external access. **Volume** objects are used for persistence data storage of Pods data. **Namespace** is a virtual cluster which allows to isolate scoped Kubernetes objects such as Pods, Persistent Volume Claims (PVCs), and Deployments. A **ResourceQuota** object provides constraints that limit aggregate resource consumption per namespace. It can limit the quantity of objects (e.g. number of NodePorts, PVCs, Pods, etc.) that can be created in a namespace by type, as well as the total amount of compute resources that may be consumed in that namespace.

### B. Slicing at the Edge

Recently, both in academy and industry, different container based edge cloud solutions are being developed to manage and orchestrate network services at the edge of the network. Open source solutions for the edge cloud are also developed [7]. In the view of enabling network slices for MEC/edge cloud services, authors of related work proposed different approaches. A good survey of state of the art on cloud native network slicing is discussed in [5], while another notable work on slice enabled MEC in 5G is discussed in [6]. Though there are many works in this regards, most of them are focused on features already available by container orchestration solutions such as Kubernetes and are not achieving the main goal of slicing, i.e., isolation and resource control. In this paper we present means to achieve slicing by extending and combining features of Kubernetes. The Kubernetes platform provides extensible Open APIs for all of its services, hence this allows us to use available services and extend them for our purpose. In this regard we combined the feature of Kubernetes Namespaces and ResourceQuotas to meet the two main requirements of network slices i.e 1) isolation which can be achieved by Kubernetes Namespaces and 2) Resource control which can be achieved by Kubernetes ResourceQuotas. Combining the two Kubernetes objects, we developed an abstraction representing *slice objects* (see Listing 1). We implemented a REST based slice management API, an Open API from the Edge Controller (Fig. 1) towards the above orchestration layers, i.e to create, update and delete slices. The Edge Controller extends the Kubernetes master API and implements the additional features sketched in Sec.II, incl. the Platform Manager per the ETSI MEC architecture [2].

A slice is a scoped cluster and is viewed as a resource allocation spanning across all worker nodes in the cluster. At the Edge Controller level we have the opportunity to see which Pod is running in which node, even if it is assigned to a specific slice. In this regards we introduce a novel concept called $\mu$-slices i.e the available budget for a slice per worker node, this enable as to flexibly control placement of Pods/CNFs to a specific $\mu$-slice without violating the total slice budget. With reference to Fig. 2, in a highway scenario with distributed nodes deployed along a geographic area, $\mu$-slices can be mapped to suitable geo-locations, hence this allows us to introduce geo-aware slices. The aggregate resource that will be assigned to each geo-slice will be equal to the total slice budget of the tenant. This concept of mapping geo-areas with $\mu$-slices allows us to easily shift resources from one $\mu$-slice to another $\mu$-slice without exceeding the overall aggregated resources admitted for a slice. Shifting resources beyond $\mu$-slices managed by an Edge Controller can be coordinated via edge orchestration layers or NFV-SO with Edge Controllers of other edge resources.

[7]KubeEdge: https://kubeedge.io/en/

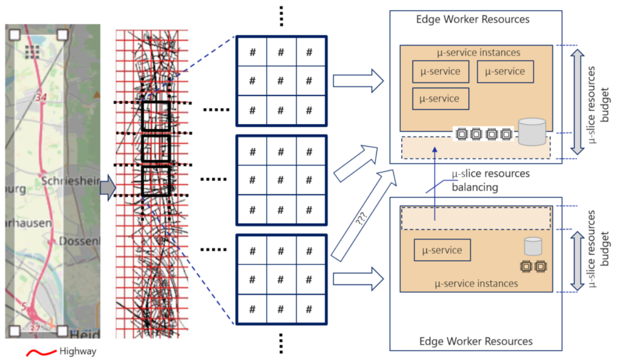

Figure 2. Leveraging demand predictions for $\mu$-slice resources management

## IV. LSTM-based Traffic Forecasting as Enabler for Proactive Edge Resources Allocations

A proper allocation of resources in one or multiple Edge Nodes or instantiated slices is fundamental for the overall performance of the mobile system. The proper amount of resources depends on the number of requests coming from the mobile users consuming the edge services and the evolution of such requests in time. Being able to anticipate such users behavior is the key enabler to proactively set edge resources accordingly [7].

In this work, we leverage Recurrent Neural Networks (RNNs) for time series forecasting. The time series we consider is composed by the number of users in different locations of the network in a time interval from t-N to t (present time). For each location, we use RNNs to predict the number of vehicles at time t+1. In particular, we use Long Short Term

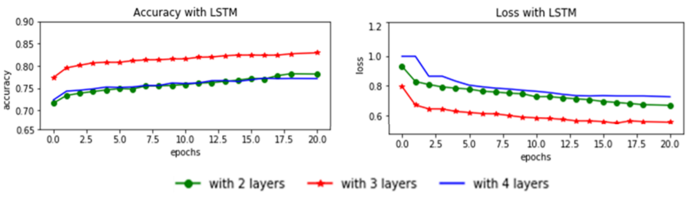

Figure 3. Accuracy and loss results when training an RNN LSTM on a real map highway and urban area traffic forecasting.

Memory (LSTM) special units to overcome RNN well-known short memory limitation [8].

LSTM is used to forecast density of vehicles in different 500x500m square cells, and this are aggregated in wider 1.5x1.5 km geo-areas, as depicted in Fig. 2. Each geo-area can then be served by one or multiple edge resources. The allocation (or modification) of such budget of resources can be based on the predicted traffic for that area. In order to take a good decision on resource allocation, a good forecast in terms of accuracy and loss is needed. Fig. 3 shows training accuracy and loss achieved through LSTM applied on the described traffic: using a 3 layer configuration, LSTM is able to provide an accuracy above 0.8, thus allowing for a good prediction of the traffic highway patterns.

In the results, highway traffic is simulated using SUMO[8]: different flows of vehicles entering and exiting the highway at different random entrances and exits are generated at various rates during a total interval of 24 hours (100veh/minute at the highest peaks). Speed of vehicles is defined with a normal distribution having mean value as 0.8 of the max street speed limit (i.e., 50 m/s) and std.dev. equal to 0.2 of said speed limit. More details on the parameters are contained in [9].

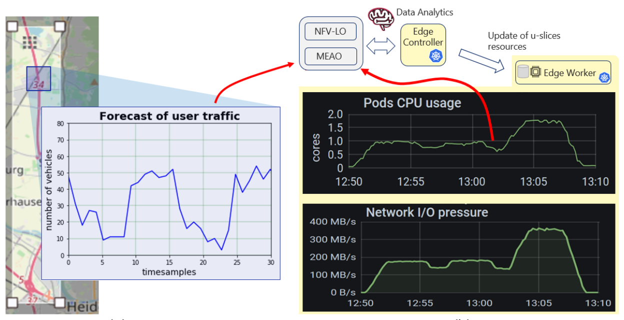

Figure 4. Example of vehicle traffic forecast over time in a geo-area (a) and CPU and Network Utilisation in an Edge Worker (b). Both information can be leveraged by the Mobile Edge Application Orchestrator (MEAO) in order to take u-slice resource distribution updates on Edge Workers.

## V. Towards Edge Automation for CAM

To prove the aforementioned concepts, we build an evaluation setup including a Kubernetes-cluster-based Edge Controller having two Edge Worker nodes. We define a network slice instance with a specific resource budget (Listing 1), comprising three pods of which two are running on one Edge Worker and the third on the other Edge Worker node. We again consider simulated highway traffic as described in Sec. IV, and generate based on it user requests with wrk[9] and Nginx[10] tools. Then, we evaluate the traffic impact in terms of edge resources consumption: in Fig. 4(b) pods CPU and Network utilisation on one Edge Worker are depicted.

The resource consumption can at some point increase because of increasing user traffic, as it happens around time

**Listing 1** Slice Descriptor (namespace and resources)

```
 ..."metadata": { "name": "slice1"},
 "spec": {
     "limits.cpu": "4",
     "limits.memory": "2048M",
     "requests.storage": "100Gi",
     "services.nodeports": "4" ....
```

13:02 in the aforementioned figure. In such case we could re-organise the overall resources: whereas the total resources for the slice instance are fixed, we can increase the quota of resources in the $\mu$-slice on the traffic peak impacted Edge Worker Node, yet reducing resources on the other Worker Nodes. Similarly, as Fig. 4(a) shows, we can update the $\mu$-slice resources for each Edge Worker directly based on the traffic forecasting: when we forecast a peak in user traffic for the specific geo-area, we can enforce through the Edge Controller a consequential re-organization of the edge resources in the different $\mu$-slices.

## VI. Conclusion

In this paper we combine the possibility to flexibly allocate $\mu$-slice resources at edge nodes with machine learning-based traffic forecasting to anticipate the clients' traffic patterns and assess edge resources demand accordingly. We show the feasibility and advantages of such an approach based on our OSS-prototype for orchestrated edges, currently under development for the 5G-CARMEN European project.

## Acknowledgment

The research leading to these results was supported by H2020-ICT-2018 grant agreement No. 825012 (5G-CARMEN).

## References

[1] 3GPP, "Management of Network Slicing in Mobile Networks; concepts, use cases and requirements," 3rd Generation Partnership Project, TS 28.542.

[2] ETSI, "Multi-access Edge Computing (MEC); Framework and Reference Architecture," European Telecommunication Standards Institute (ETSI), TS, January 2019.

[3] 5G-CARMEN, "Deliverable 4.1 - Design of the secure, cross-border, and multi-domain service orchestration platform," *H2020 5G-CARMEN Project Consortium*, 2020, online [Available]: https://5gcarmen.eu/wp-content/uploads/2020/11/5G_CARMEN_D4.1_FINAL.pdf.

[4] R. Cziva, S. Jouet, K. J. S. White, and D. P. Pezaros, "Container-based network function virtualization for software-defined networks," in *2015 IEEE Symposium on Computers and Communication (ISCC)*, 2015, pp. 415–420.

[5] S. D. A. Shah, M. A. Gregory, and S. Li, "Cloud-native network slicing using software defined networking based multi-access edge computing: A survey," *IEEE Access*, vol. 9, pp. 10 903–10 924, 2021.

[6] A. Ksentini and P. A. Frangoudis, "Toward slicing-enabled multi-access edge computing in 5g," *IEEE Network*, vol. 34, no. 2, pp. 99–105, 2020.

[7] S. Ntalampiras and M. Fiore, "Forecasting mobile service demands for anticipatory mec," in *2018 IEEE 19th International Symposium on" A World of Wireless, Mobile and Multimedia Networks"(WoWMoM)*. IEEE, 2018.

[8] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[9] U. Fattore, M. Liebsch, B. Brik, and A. Ksentini, "Automec: Lstm-based user mobility prediction for service management in distributed mec resources," in *Proceedings of the 23rd International ACM Conference on Modeling, Analysis and Simulation of Wireless and Mobile Systems*, 2020.

[8]SUMO simulation tool: https://www.eclipse.org/sumo/

[9]wrk - a HTTP benchmarking tool: https://github.com/wg/wrk

[10]Nginx: https://www.nginx.com/