M^{2}EC: A Multi-tenant Resource Orchestration in Multi-access Edge Computing Systems

Lanfranco Zanzi, Fabio Giust, Vincenzo Sciancalepore
NEC Laboratories Europe, Germany
E-mail: {name.surname}@neclab.eu

***Abstract*—Multi-access Edge Computing (MEC) is envisioned as a key technology in the 5G landscape, able to bring computing capabilities to the edge of the network, closer to the end users. This would help to fulfill the next generation network requirements in terms of low latency and high bandwidth. ETSI has started specifying MEC as an operator-owned system to run third party's applications, which can leverage the benefits inherent to the MEC environment for added value services to the end users.**

In this paper we explore a novel paradigm for third parties to access the MEC system: renting part of MEC facilities leveraging on the network slicing paradigm to expand the business opportunities for both the system provider and the MEC tenants. We introduce the concept of *MEC broker* as an entity exposing administration and management capabilities while handling heterogeneous tenant privileges. Our concept is validated by developing an orchestration solution, namely M^{2}EC, to optimally allocate requested resources in compliance with the tenants service level agreements.

I. Introduction

The convergence of Information Technology and networking finds a realization when it comes to Multi-access Edge Computing (MEC). It is widely recognized as a promising technology to bring cloud computing capabilities to the edge of network, where low latency and high bandwidth can be exploited by cloud applications in order to deliver added-value services to the end users. Targeted use cases include tactile Internet, augmented and virtual reality, live streaming, etc., and also those belonging to specific vertical industry segments, as industrial automation, eHealth, automotive, etc.

The European Telecommunications Standards Institute (ETSI) has chartered the Industry Specifications Group[1] precisely to define a multi-vendor standardized MEC system to allow third party software providers to install their applications in the network operator's premises. According to the ETSI architecture [1], an *MEC provider* (e.g., a mobile network operator) owns the system that comprises a set of IT hosts and the management entities, building the concept of Infrastructure as a Service (IaaS). Third party software providers (i.e., the *MEC tenants*) deliver their application package(s) to the provider as responsible for the deployment of such application instance(s) in the MEC hosts, configuring the appropriate parameters and traffic rules, and for the policies enforcement to provide specific tenant Service Level Agreements (SLAs).

[1]Please refer to http://www.etsi.org/technologies-clusters/technologies/multi-access-edge-computing for further information.

Beyond the IaaS model, network operators are exploring novel business means to monetize their infrastructure by offering *Network Slices* to external tenants [2]. From the MEC perspective, a slice is envisioned as a technological opportunity for the tenant to operate and manage the MEC system according to different privilege levels while gaining more control over the delivered service. This would result in an advanced multi-tenancy multi-access edge computing architecture, namely M^{2}EC. An example is represented by a Mobile Virtual Network Operator (MVNO) willing to acquire a slice of MEC to expand its business model and operate on the network management in order to tailor the system according to its dynamic requirements.

The MEC slicing concept enables the infrastructure provider to facilitate a set of heterogeneous privileges to multiple tenants. However, when different tenants operate on a shared infrastructure, efficient mechanisms must be in place to validate upcoming tenant requests and policies and to enforce them while avoiding conflict states. In this context, our contributions may be summarized as follows.

- Introducing the *MEC Broker* from an architectural viewpoint as an entity capable of exposing heterogeneous administration privileges to MEC tenants, of processing tenant requests, and of resolving resource contentions.
- Devising an *Orchestration mechanism* compliant with the ETSI-defined operations, able to fulfill the tenant requests and to avoid SLA violations.
- Defining an *Optimization Framework* that supports the MEC slicing paradigm pursuing the infrastructure resource efficiency maximization while accounting both end-to-end application delay requirements and platform computation/storage capabilities.

The rest of the paper is structured as follows: Section II provides a general overview of the ETSI MEC system and a state-of-the-art discussion. Section III introduces the conceptual architecture and means to support network slicing in MEC systems. The orchestration solution is described in Section IV by means of a model to compute resources allocation, whereas Section V validates the optimization framework through exhaustive simulations. Finally, Section VI concludes the paper.

II. Background

We present a network slicing support mechanism built on top of the ETSI-defined MEC architecture [1], which simpli-

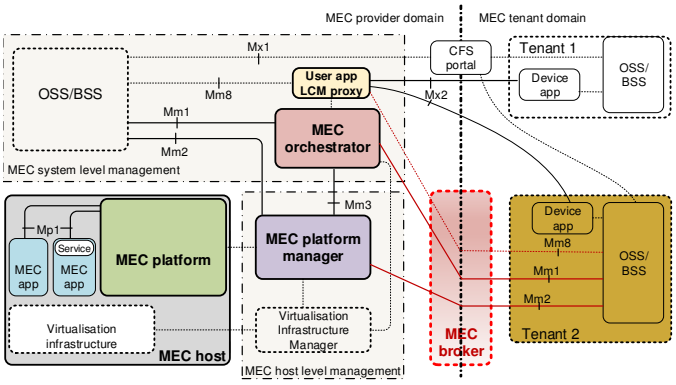

Fig. 1. Simplified Multi-access Edge Computing (MEC) architecture enhanced with the MEC broker. The diagram shows how different tenants can access the system via the legacy mechanism (Tenant 1) and the one proposed by the present solution (Tenant 2).

fied version is depicted in Fig. 1 whereas a brief description is given below.

In the service model envisioned by the MEC architecture, the tenants deliver their application software and the descriptor files to the system provider. The MEC host is then the IT infrastructure where such applications run as virtual machines (VMs). The MEC platform sitting in each host enables the applications by providing them access to the MEC service API endpoints, handling DNS procedures and enforcing the traffic rules and policies on the data plane [3].

In order to get the applications running in the desired location of the network, the MEC provider's Operations Support System (OSS) is the highest level management entity to accomplish the task, upon the requests coming from an MEC tenant. This is reflected on top of Fig. 1, where *Tenant 1* is represented as an entity similar to an OSS/BSS (Business Support System) which uses the Customer Facing Services (CFS) portal in order to request the instantiation and termination of an MEC application. The MEC provider's OSS receives the requests from the CFS portal and operates the MEC orchestrator and the MEC platform manager to fulfill them. Additionally, the MEC system allows applications running in the user's device to interact with the MEC system to perform the requests through the User app LCM proxy, which is connected to the MEC provider's OSS and MEC orchestrator to validate and satisfy the requests. It is safe to assume that also the user application logically belongs to the tenant, so that Tenant 1 using the MEC legacy system is allowed to interact with the MEC system only through the CFS portal and User app LCM proxy.

The role of the Tenant 1 is thus limited to controlling the applications logic (e.g., through remote access to the applications backports) once the application is up and running, whereas the MEC provider is responsible for the instance configuration and management, as per the following non-exhaustive list of operations, in fulfillment of the agreed SLAs and of the MEC service providers own policies and capabilities:

- Application placement, i.e., the set of available MEC hosts where the application is installed and executed;
- Management of the application-assigned networking, computing and storage resources;
- Application LCM, including bootstrapping, termination, scaling in/out and up/down, migration (this latter assumes also validation and enforcement of the policies for application migration to other MEC hosts);
- DNS and traffic rules configuration in order to provide the appropriate connectivity.

While the IaaS model described above meets the expectations of several tenants, an MEC slice creation and management concept of different tenants attracts additional stakeholders having the capabilities and willing to gain full control over the delivered service. In Section III we propose the architectural enhancements to MEC and the key concepts to enable MEC slicing. In the following paragraph we showcase some prior art on the topic.

A. Related Work

To the best of our knowledge, we pioneer the slice resource allocation framework and practical mechanisms in an ETSI-compliant MEC system. However, the network slice broker entity has been initially introduced in [4] with the objective of solving the admission and control problem of slice requests in mobile Radio Access Network (RAN) facilities. Performance evaluations there show promising results in terms of low SLA violations even in dynamic scenarios [5]. Bringing this entity in MEC environments opens the possibility to exploit results from other research domains, given the parallelism with bin packing problem in cloud computing and virtual function placement problem in virtual network embedding fields. In cloud computing context, a wide set of optimization criteria can be defined [6]. Energy consumption or average latency minimization, Quality of Experience (QoE) maximization as well as optimization of the number of migrations. All these metrics are also suitable for MEC systems, even if they are less performing and consume less energy due to limited capabilities with respect to cloud data centers. The authors of [7] address the VNF placement problem by considering the highly dynamic nature of cloud systems and by modeling requests as a continuous stream. Their solution considers two steps. The first consists in a continuous deployment decisional process, which is followed by a placement re-optimization phase that includes migrations. A way to optimize migrations is to consider consolidation of resources. [8] deals with initial VM placement including spatial and temporal awareness for consolidation purposes based on the forecast of resource demand in time. Similarly to our work, the authors of [9] and [10] present VNF placement and provisioning optimization strategies over edge and cloud infrastructure taking into account Quality of Service (QoS) requirements. Their objective is to solve the trade off between optimization of resource utilization and the minimization of SLA violations.

III. A Broker to Support Slicing in MEC

In the previous section a legacy mechanism to access the MEC system by third parties and its limitations are shown. Advanced tenants willing to decide on the application LCM on their own, should be granted access to the management

entities that are visible from the MEC provider's OSS, i.e., the MEC orchestrator, the MEC platform manager and the User app LCM proxy.

In this paper, we thence propose the *MEC broker*, as a logically entity able to expose to a tenant's OSS the interfaces that connect the MEC provider's OSS to the entities mentioned above (namely, the interfaces supported by MEC reference points Mm1, Mm2 and Mm8 [1], [11], [12]). In Fig. 1, one can see that *Tenant 2* is enabled with enhanced operations through the proposed MEC broker, as compared to Tenant 1 which accesses the MEC system with legacy mechanisms only.

By exposing the management interfaces to multiple tenants, the MEC broker enables them to manage the same system even simultaneously, thus it must ensure the consistency of the policies, configuration items and commands issued by the tenants. Therefore, the MEC broker assigns tenants with *privileges* and *priorities*. Privileges refer to the set of allowed actions that a tenant is authorized to perform, which map directly to the usage of the MEC interfaces. Priorities refer to the validity in time of a privilege, and how commands issued by a tenant take precedence over those by the others.

Conflicts may occur when shared resources are simultaneously used or if the MEC system is running out of resources. The MEC broker will efficiently try to reduce the number of conflicts by promptly balancing the MEC tenants applications over available MEC hosts. This is strictly related to the privileges issued to running MEC tenants and might be needed an optimal admission control in charge of filtering incoming MEC slice requests based on the current availability.

In order to carry out the operations above described, the MEC broker is logically equipped with functional elements that:

- Implement the exposure of the interfaces that are transported over the MEC reference points connected to the OSS entity, namely Mm1, Mm2 and Mm8;
- Grant, revoke, modify and check privileges and priorities, including the communication to the tenant to enable such operations; This block may consist of a login-based procedure through which a tenant acquires privileges and priorities, asks for updates and queries the status. A list of available privileges is provided by the MEC broker, which may update it and forward to MEC tenants (upcoming or currently connected).
- Record all the instructions issued by tenants over the exposed MEC reference points in order to validate and execute them. When any of the MEC operations is requested by the tenant, the priority associated to the privilege is looked up. The look-up determines if the operation is actually granted to the tenant and the conflicts associated to the operation (e.g., installing a DNS record with an already used IP address or domain name) are evaluated. If the check is successful the operation can be executed. Otherwise, if a conflict is detected, the system generates an alarm to all the lower priority tenants, which are automatically disabled from executing such operations.

From a deployment point of view, the MEC broker can be realized in different ways, e.g., *i)* as an additional entity, like that depicted in Fig. 1, *ii)* by extending the capabilities of the MEC Orchestrator, the MEC platform manager and the User app LCM proxy, *iii)* by augmenting the MEC providers OSS or the CFS portal.

The architectural enhancements proposed in this paper enable tenants to access an MEC system with additional control over the MEC resources as compared to the legacy mechanism. In the following sections, we describe and validate an orchestration system able to allocate MEC resources when requested using both the legacy and our mechanism.

IV. Multi-tenant Resource Orchestration

The role of the MEC orchestrator is to properly instantiate the tenant applications into the set of MEC hosts fulfilling the functional requirements of the applications and the agreed SLAs. Functional requirements might include virtual resources such as computing burden, storage and network throughput, as well as the dependency to particular *MEC services*. An MEC service is a specific application in an MEC host able to expose an API to other applications providing enhanced information, e.g., radio conditions [13] or location of users [14]. Some services are built-in within the MEC platform whereas others may be installed on-demand. In addition, MEC applications run in fulfillment with tenant SLAs, e.g., the maximum tolerable delay experienced by the application client running in the user device, the maximum number of application instances running simultaneously in the MEC system, or a list of granted hosts for a specific application due to regulatory limitations.

In light of the above considerations, we define three different categories of MEC tenants based on heterogeneous application requirements and given privileges:

Basic. Tenants request to run one or more instances of the application on different hosts, with loose delay requirements and in absence of management privileges;

Premium. Tenants request to run one or more instances of the application on different hosts with *stringent* delay requirements and in absence of management privileges;

Gold. Tenants request direct access to an isolated slice of the MEC platform, which includes both management privileges and low-delay guarantees. This slice provides a guaranteed MEC applications deployment onto specific MEC hosts.

The orchestration system aims at allocating needed resources to run tenant applications while fulfilling application delay requirements, i.e., to accommodate applications onto available hosts. We devise an orchestration algorithm in charge of finding the optimal placement while pursuing the overall MEC hosts resources minimization. This opens up new opportunities for admitting additional MEC tenants (or MEC slices) and, in turn, increasing the overall system revenues.

A. Scenario Characterization and System Model

The MEC system is described as the set of deployed hosts H_i, where $i \in \mathcal{I} := \{1, \ldots, I\}$, connected to each other through provider backhaul links as per the Mp3 MEC reference point [1]. Thus, any host pair (H_i, H_j), $i, j \in \mathcal{I}$, is logically connected via a link which associated overall latency is $\delta_{i,j}$, regardless of the actual number of hops. In addition,

each host is equipped with fronthaul links for user access (i.e., towards the base stations) which are modeled following the same principle. Thus, the variable λ_i accounts for the average delay between the users connected to host H_i through all its access link and the host itself (as shown in Fig 2). Given the purpose of our model to guarantee delay constraints, for the sake of simplicity we assume that each link has enough capacity to satisfy traffic requirements. Moreover, each host H_i is characterized by its computing resources, synthesized by the *total capacity* parameter c_i. Thus, each host can be described as $H_i = \{c_i, \lambda_i, \boldsymbol{\delta_i}\}$ where $\boldsymbol{\delta_i}$ comprises delays $\delta_{i,j}, \forall j \neq i$. Services offered by the MEC system are modeled as set S_w where $w \in \mathcal{W} := \{1, \ldots, W\}$, and each service S_w consumes s_w resources from the host[2].

The infrastructure described above shall accommodate the set of applications A_k requested by tenants, where $k \in \mathcal{K} := \{1, \ldots, K\}$. Without loss of generality, hereafter we refer to a k-th tenant through its application A_k. Although the admission and control procedure is out of the scope of the paper, it is assumed that such a mechanism is in place and translates the incoming tenant requests into the following parametrization associated to the requested application A_k:

- a_k application's processing consumption;
- Δ_k maximum tolerable end-to-end delay between the application and the user consuming such an application;
- $b_{k,w}$ list of required services for each application.

Additionally, we assume each tenant asks for the deployment of its application in one or more hosts, thus requests are modeled as a set of binary variables $z_{k,i} \in \{0, 1\}$, where $z_{k,i} = 1$ if A_k is requested on host H_i, and 0 otherwise.

Overall, the above information allows to optimally define the placement strategy for applications and services over MEC facilities in a multi-tenant scenario. Obviously, the procedure becomes more challenging as the number of hosts and requirements stiffness increases.

B. MEC Slicing Problem

Multiple instances of the same application can run over different hosts in the network. However, given the limited capacity of MEC hosts and assuming a non-uniform distribution across the network, it could be useful from the provider's point of view to migrate some applications. For instance, an overloaded host can be offloaded by migrating some running instances to another location [15]. MEC systems could further benefit from migrations if we consider that different applications may require the same services. In particular, *Consolidation* of spread applications allows for an overall processing capacity utilization reduction and consequent operational cost savings [16]. Moreover, the saved capacity could be engaged to admit more requests in the future and to increase the acceptance rate given the same physical network, thus, from an economical standpoint, rising revenues.

[2]Please note that we assume a constant utilization of computing resources, irrespective of the actual load. A more accurate model should account for the consumption as a function of the load, being the derivation of such function a complex modeling problem by itself, which is out of scope of the present study.

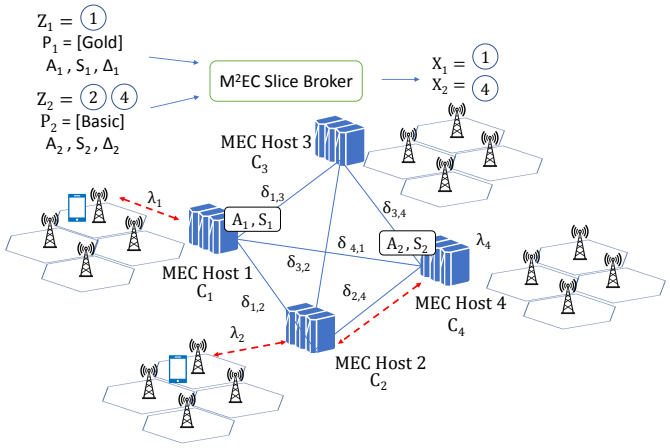

Fig. 2. M²EC Slicing Architecture

Request arrivals/departures occur every time a tenant requests to modify locations and/or privileges or deploy new applications. Upon receiving the slice requests set, the provider has to deal with the applications placement problem, pursuing the objective of overall resource utilization minimization while honoring the past agreed guarantees. Fig. 2 depicts the workflow in case of *K*=2, *W*=2 and *I*=4. In this example, two tenants ask for the deployment of their application on specific hosts providing latency and service requirements. The tenant with higher privileges asks for the application deployment on the first host while the other tenant, with basic privileges, on the second and fourth hosts. At the end of the decisional process, *gold*-type requests are completely satisfied whereas *basic*-type requests are properly mapped by the M²EC system so as to guarantee the delivery of the service while saving MEC resources. In particular, the second application has loose delay requirements so it can be enabled in a less congested location, e.g., host H_4, without affecting the final service delivery. In this case, mobile users under the coverage area of host H_2 can access the service through host H_4. We can formulate our problem as the following.

Problem 1 (MEC Slicing Problem):

$$\min_{\boldsymbol{x},\boldsymbol{y}} \sum_{i\in\mathcal{I}} \Big(\sum_{k\in\mathcal{K}} a_k x_{k,i} + \sum_{w\in\mathcal{W}} s_w y_{w,i} \Big) \tag{1a}$$

$$\text{s.t.} \sum_{k\in\mathcal{K}} a_k x_{k,i} + \sum_{w\in\mathcal{W}} s_w y_{w,i} \leq c_i, \qquad \forall i \in \mathcal{I}; \tag{1b}$$

$$z_{k,j}(\lambda_j + \delta_{i,j}) \leq \Delta_k x_{k,i} + M(1 - t_{k,i,j}), \quad \forall k \in \mathcal{K}, \forall i, j \in \mathcal{I}; \tag{1c}$$

$$z_{k,i} p_k - x_{k,i} \leq 0, \qquad \forall k \in \mathcal{K}, \forall i \in \mathcal{I}; \tag{1d}$$

$$b_{k,w} x_{k,i} - y_{w,i} \leq 0, \qquad \forall i \in \mathcal{I}, \forall k \in \mathcal{K}, \forall w \in \mathcal{W}; \tag{1e}$$

$$\sum_{i\in\mathcal{I}} t_{i,j,k} \geq 1, \qquad \forall k \in \mathcal{K}, \forall j \in \mathcal{J}. \tag{1f}$$

Decision Variables: The decision variable $x_{k,i} \in \{0,1\}$ denotes whether an incoming tenant request for application A_k is placed on host H_i. The decision variable $y_{w,i} \in \{0, 1\}$ establishes whether the service S_w is enabled on host H_i. Finally, $t_{i,j,k} \in \{0, 1\}$ fictitiously models the choice to migrate the application A_k from H_i to H_j.

Objective Function: The main goal is the definition of an optimal MEC applications placement, which allows the coexistence of heterogeneous tenants while minimizing the overall resource consumption. The result can be *a-posteriori* translated into operational costs reduction policies or used to efficiently drive the current admission and control strategy by increasing the request acceptance ratio.

Constraints: Eq. (1b) represents a capacity constraint that relates application and service consumptions with the hosts capability. Eq. (1c) sets the maximum delay budget for each application and destination host. Tenant requests are represented by the variable $z_{k,j}$ as explained in Section IV-A, which takes into account the willing of the tenant to deploy the application A_k on a specific MEC host H_j. However, such a tenant might not have the privileges ($p_k \in \{0, 1\}$) to demand for a guaranteed MEC applications deployment, and thus its instance might be automatically migrated to a more convenient location. With Eqs. (1c) and (1f), we assure that at least one delay request from each tenant is satisfied during the decisional process by exploiting the *Big M Method* [17] and the fictitious variable $t_{i,j,k}$. This also prevents from applying the straightforward solution of placing no tenant request in our MEC system. The value of M must be chosen sufficiently large so that the fictitious variable would not be part of any feasible solution (for e.g., $M = 1000$). Application consolidation and migration are applied based on latency values of MEC links accounting for a delay cost $\delta_{i,j}$ between hosts H_i and H_j. As introduced before, tenants belonging to different categories are provided with diverse privileges. This is taken into account by Eq. (1d), which ensures that *gold*-type requests (p_k = 1) will be entirely satisfied. Last, Eq. (1e) enables the concurrent deployment of selected services required by all applications ($b_{k,w}$) running on specific hosts.

V. Solution Validation

We choose a real network topology from [18] for evaluation purposes. In particular, such network deployment, namely GARR, is composed by 37 hosts spread over the italian territory and more than 80 edges connecting them. Given the set of edges and node locations, the delay matrix with values $\delta_{i,j}$, $\forall i, j \in \mathcal{I}$, is easily obtained running the Dijkstra's algorithm. Let $\bar{\delta}$ and δ_{max} define respectively the average and maximum value of such delays. Without loss of generality, host capacities are equally distributed and normalized to a unitary value. The computational requirements for each application and service are expressed with respect to a fixed value γ, which represents $1/100$ of the single host capacity, as resumed in Table I.

These values are small enough to fit into hosts, as it would be after the execution of the admission and control process. The relationship between services and applications modeled through the binary variable $b_{k,w}$ is obtained randomly at runtime. Given the lack of comparable MEC slicing solutions in the literature, we firstly evaluate our proposal against the baseline approach (*Legacy*) of placing the application and relative services exactly in the MEC hosts where each tenant demands. We run our simulations combining MATLAB and

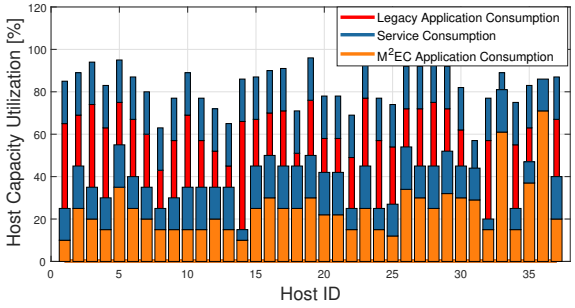

Fig. 3. Host capacity utilization for K=40 and W=4

Optimization Programming Language (OPL), together with the optimization engine CPLEX. Fig. 3 shows the host utilization for a set of requests distributed among the available categories according to 40%, 40%, 20% ratios, respectively. The *legacy* solution results in an overall increase of resource utilization due to the unavoidable concurrent deployment of the same services and applications, even on nearby hosts. In the M^{2}EC scenario, the MEC applications deployment privilege is guaranteed only to *gold* tenants. It can be noticed that M^{2}E allows for a smarter deployment of resources accounting for services consolidation and leading to 40% of resources saving.

We investigate the MEC host capacity savings in case of variable delay requirements. By denoting with λ_{min} and λ_{max} the minimum and maximum fronthaul delay, respectively, the application delay requirement Δ_k cannot be $\Delta_k < \lambda_{min}$. We then let Δ_k vary as a uniform random variable within the interval $[\lambda_{min}, \bar{\delta} + \alpha]$, where $\alpha \in [0, \lambda_{max} + \delta_{max} - \bar{\delta}]$ represents a tunable parameter aimed to increase the random interval up to the largest theoretical delay, $\lambda_{max} + \delta_{max}$. This setup provides heterogeneity in the incoming requests since both mean value and variance of Δ_k increase during the simulations. Fig. 4 shows the average MEC system utilization for increasing values of α. It can be noticed that with a Δ_k distribution close to λ_{min}, the overall capacity utilization is maximized, while it starts exponentially decreasing as the delay interval augments. With more stringent delay requirements, the possibility of finding a suitable host close enough to satisfy the incoming application delay request dramatically decreases. As a consequence, the same application must be deployed on multiple platforms without any consolidation opportunity. The scenario becomes less challenging as the delay requirement distribution spreads over a bigger interval. This further motivates the presence of heterogeneous tenant classes, as the stringent delay requirement deeply impacts on the general system consolidation capabilities.

Finally, Fig. 5 shows the impact of an increasing number of *gold*-type tenants on the system. The number of running

TABLE I
Tenant Categories

Tenant Category	Basic	Premium	Gold
Δ_k [ms]	[100, 150]	[50, 100]	[20, 50]
p_k	0	0	1
a_k	2γ	4γ	5γ
s_k	5γ		

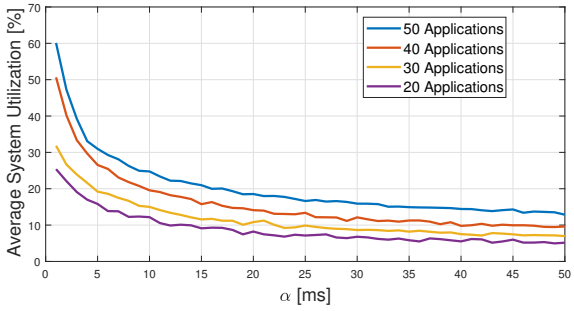

Fig. 4. Delay impact on consolidation capabilities.

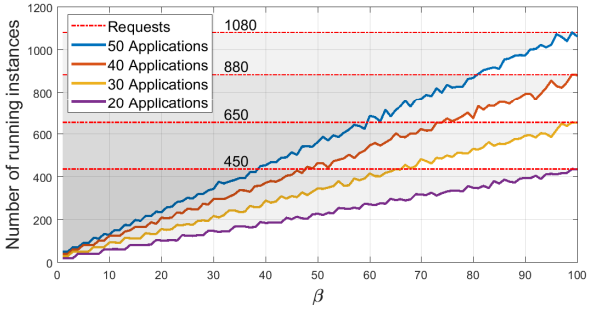

Fig. 5. Management privileges impact on consolidation capabilities.

application instances monotonically increases according to β, which represents the percentage of users belonging to the privileged category. The highlighted area between the curve of running instances and the curve of the requests is a measure of the M^{2}EC impact on the placement decisions with respect to the *legacy* straightforward solution of setting the instances only where demanded. Once again, M^{2}EC capabilities provide significant gains in terms of resource utilization savings, even with an increasing number of tenant requests. The number of running instances is minimized when $\beta = 0$ (all tenants belong to the *basic*/*premium* category) and maximized when $\beta = 100$ (all tenants belong to the *gold* category). During the admission and control phase, it must be taken into account that a majority of *gold*-type tenants limit the consolidation capability of the system and potentially interfere with the fulfillment of other agreed SLAs. Overall, our findings provides a guideline for the infrastructure providers: they might define ad-hoc solutions to prevent this issue, for instance, with different pricing labels to compensate the limited resources assignment flexibility.

Acknowledgments

The research leading to these results has been partially supported by the H2020-MSCA-ITN-2015 framework under grant agreement number 675806 (5G-AuRA) and by the European Unions Horizon 2020 research and innovation program under the grant agreement number 671584 (5G-NORMA).

VI. Conclusions

The key-enablers of 5G networks design are identified as Multi-Access Edge Computing (MEC) and Network Slicing capabilities, driven by the impelling need to provide high bandwidth as well as real-time access to mobile users in an isolated manner. In this paper we have introduced the figure of the *MEC Broker* and proposed an orchestration solution, namely M^{2}EC, to deal with the concurrent deployment of MEC applications in multi-tenancy environments, with the objective of minimizing the overall capacity utilization exploiting applications consolidation over different MEC hosts. Considering the overall MEC system economy, our analysis shows significant benefits provided by the introduction of advanced resources allocation mechanisms into the slice management. This enables costs savings while providing ad-hoc solutions for external tenants willing to place their services over edge computing systems.

To summarize, i) we proposed an architectural amendment to include the *MEC Broker* entity into the ETSI MEC standard systems, ii) we formulated an optimization placement problem pursuing the applications consolidation, which, in turn, aims at computational consumption minimization, iii) we empirically shown that our M^{2}EC outperforms the legacy approach where applications are placed trivially in the required MEC hosts.

References

[1] ETSI MEC ISG, "Mobile Edge Computing (MEC); Framework and reference architecture," ETSI, DGS MEC 003, April 2016.
[2] P. Rost *et al.*, "Network slicing to enable scalability and flexibility in 5G mobile networks," *IEEE Communications Magazine*, May 2017.
[3] ETSI MEC ISG, "Mobile Edge Computing (MEC); Mobile Edge Platform Application Enablement," ETSI, DGS MEC 011, July 2017.
[4] K. Samdanis, X. Costa-Pérez, and V. Sciancalepore, "From network sharing to multi-tenancy: The 5G network slice broker," *IEEE Communications Magazine*, vol. 54, no. 7, pp. 32–39, Jul. 2016.
[5] V. Sciancalepore *et al.*, "Mobile traffic forecasting for maximizing 5G network slicing resource utilization," in *IEEE INFOCOM*, May 2017.
[6] R. Cohen *et al.*, "Near optimal placement of virtual network functions," in *IEEE INFOCOM*, Apr. 2015, pp. 1346–1354.
[7] N. M. Calcavecchia, O. Biran, E. Hadad, and Y. Moatti, "VM placement strategies for cloud scenarios," in *5th International Conference on Cloud Computing*. IEEE, Dec. 2012, pp. 852 – 859.
[8] Liuhua Chen and Haiying Shen, "Consolidating Complementary VMs with Spatial/Temporal-awareness in Cloud Datacenters," in *IEEE INFOCOM*, Apr. 2014, pp. 1033–1041.
[9] F. B. Jemaa, G. Pujolle, and M. Pariente, "QoS-aware VNF Placement Optimization in Edge-Central Carrier Cloud Architecture," in *Global Communications Conference (GLOBECOM)*. IEEE, Dec. 2016.
[10] M. F. Bari, S. R. Chowdhury, R. Ahmed, R. Boutaba, and O. C. M. B. Duarte, "Orchestrating virtualized network functions," *IEEE Transactions on Network and Service Management*, 2016.
[11] ETSI MEC ISG, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 1: System, host and platform management," ETSI, DGS MEC 010-1.
[12] ——, "Mobile Edge Computing (MEC); Mobile Edge Management; Part 2: Application lifecycle, rules and requirements management," ETSI, DGS MEC 010-2, July 2017.
[13] ——, "Mobile Edge Computing (MEC); Radio Network Information API," ETSI, DGS MEC 012, July 2017.
[14] ——, "Mobile Edge Computing (MEC); Location API," ETSI, DGS MEC 013, July 2017.
[15] Weimin Qiu, Zhuzhong Qian, Sanglu Lu, "Multi-objective virtual machine consolidation," in *10th International Conference on Cloud Computing*. IEEE, Apr. 2017, pp. 1346–1354.
[16] Z. Huang and D. H. Tsang, "SLA guaranteed virtual machine consolidation for computing clouds," in *International Conference in Communications (ICC)*. IEEE, 2012, pp. 1314–1319.
[17] G. Igor, N. S. G., and S. Ariela, *Linear and Nonlinear Optimization*. Society for Industrial Mathematics, 2009.
[18] S. Knight *et al.*, "The internet topology zoo," *IEEE Journal on Selected Areas in Communications*, vol. 29, no. 9, pp. 1765–1775, 2011.