# DEEM: Enabling Microservices via DEvice Edge Markets

Argyrios G. Tasiopoulos, Onur Ascigil, Sergi Rene, Michał Król, Ioannis Psaras, and George Pavlou
Department of Electronic and Electrical Engineering, University College London, UK.
Email: {argyrios.tasiopoulos, o.ascigil, s.rene, m.krol, i.psaras, g.pavlou}@ucl.ac.uk

***Abstract*—Native applications running over handheld devices have an irreplaceable role in users' daily activities. That said, recent studies show that users download on average zero new applications on monthly basis, which suggests that new apps can face discoverability issues. In this work, we aim for a web-based, download/installation-free access to native application features through microservices ($\mu$Services) that are shared between user devices in a peer-to-peer (P2P) manner. Such a P2P approach is self-scalable and requires no investment for $\mu$Service deployment, unlike mobile edge computing or Data Centre.**

**We introduce DEEM, a DEvice Edge Market design that enables device-hosted $\mu$Services to end-users. In DEEM, $\mu$Service-based markets act as rendezvous points between available $\mu$Service instances and clients. DEEM ensures the *i*) assignment of instances to the users that value them the most, in terms of QoS gain, and *ii*) devices' income maximisation. Our evaluation on synthetic settings demonstrates DEEM's capability in exploiting the pool of device instances for improving the application QoS in terms of latency.**

## I. Introduction

The growth of native mobile applications has been unprecedented with millions currently offered at app-stores [2]. According to recent studies, the number of annual application download is expected to exceed 260 billion in the years to come, following hardware upgrades that enable new app features [3]. That said, new applications face discoverability challenges when competing against a plethora of existing options in the app-stores: users typically interact with 10 apps per day (or 30 apps per month) while downloading on average zero new apps per-month [1].

The discoverability challenge of the new apps has lead to joint efforts of developers and mobile operating systems to focus on technologies that enable effortless access of applications, *i.e.,* by skipping the download and installation steps, as in the case of *Progressive Web Apps* (PWAs) [28]. PWAs are web-based apps designed to provide the functionality of native (*i.e.,* locally running) applications through a distributed "client-server" model: browsers and service workers (a virtual proxy that enables caching and offline access to app features) running at the client's end interact through the network with *microservices* ($\mu$Services) [15] specialised in providing a subset of app features using a Function-as-a-Service model at the server-side [20]. Major applications who have successfully jumped on the PWA bandwagon include Twitter, Expedia, and Alibaba, to name a few.

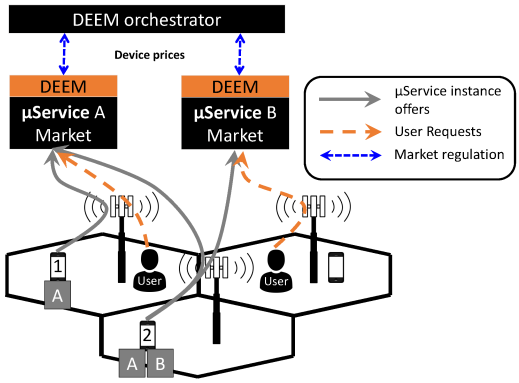



Fig. 1: $\mu$Service-specific Markets

However, the success of PWAs relies on the "smooth" (*i.e.,* low-latency and reliable) interaction between service workers at the client and *microservices* ($\mu$Services) [15] running at the server side [33]. This makes distant Data Centres unfit for the purpose of $\mu$Service deployment due to the long round-trip times (RTTs) [22] and single point-of-failure [32]. On the other hand, *Mobile edge computing (MEC)* [30] promises low-latency access to computing resources deployed nearby the users at cellular base stations or other edge nodes such as routers or cloudlets. However, MEC requires vast capital investment to provide over-provisioned resources capable of dealing with the peak user demand.

Instead, we consider a self-scaling, peer-to-peer (P2P) $\mu$Service sharing (*i.e.,* leasing) approach, whereby a pool of mobile devices that have installed $\mu$Service instances act as servers that provide services to other client devices. As opposed to a MEC deployment with costly in-network resources, our approach exploits virtualisation techniques, such as containers [18] and unikernels [23]), which enable the deployment of $\mu$Services directly on user mobile devices. That said, $\mu$Service sharing occupy computation and storage resources while depleting the energy of server devices. Hence, in this work we argue for the need of a *market-based solution that incentivises devices to lease their $\mu$Services*, similar to other emerging commercial projects that aim to provide a P2P computing infrastructure [7], [8], [13]. The vision is to create a decentralised system, where server devices are incentivised to perform application specific work and automatically receive rewards upon tasks completion.

We introduce **DEEM**: a **DE**vice **E**dge **M**arket framework

that enables the delivery of $\mu$Services, hosted in mobile devices, to end users at an improved QoS expressed as a decreasing function of the derived RTT. DEEM is a platform consisting of $\mu$Service-specific markets deployed over the edge of the network, which act as rendezvous points for user requests and $\mu$Service device instances, as shown in Fig. 1.

In DEEM, user requests are forwarded to the corresponding $\mu$Service market, while devices lease their resources by participating simultaneously in multiple $\mu$Service markets given their installed instances. DEEM assigns user requests to device $\mu$Service instances via Vickrey-English-Dutch (VED) multi-item unit demand auctions [12] that assure that each device instance is assigned to the user that values it the most at the minimum possible price. At the same time, DEEM orchestrates the $\mu$Service-specific markets by exchanging achievable price of device instances between different markets and iteratively updating their reservation price, *i.e.,* the minimum price at which an instance is offered, to maximise the device income subject to its service capabilities.

The main technical contributions of this paper are as follows:

1) We study the problem of enabling $\mu$Service instances to end users from an economic perspective.
2) We propose a $\mu$Service-specific market approach, deployed over the edge of the network, which acts as a rendezvous point between offered device instances and user requests.
3) We introduce DEEM mechanism, a P2P framework for assigning instances to the users that value them the most, while maximising the compensation of devices' participation over different markets.

The rest of the paper is organised as follows. In Section II we describe the system model and VED auctions preliminaries. In Section III we detail the DEEM mechanism. Section IV provides the performance evaluation results. Finally, we present the related work in Section V before concluding in Section VI.

## II. System Model and Preliminaries

In this section, we consider the state of a market before the execution of the assignment (reassignment) mechanism that will define the users served by each device, presented is details in Section III.

### A. System Model

Let $\mathcal{S} \triangleq \{1, 2, ..., S\}$ be a set of $\mu$Services, where each $s \in \mathcal{S}$ is provided by a set of mobile devices $\mathcal{D}_s$. We denote with $\mathcal{D} = \{1, ..., D\}$ the set of devices participating over the application specific markets, willing to serve $\mu$Services requests, where each device $d \in \mathcal{D}$ is equipped with a set of $\mathcal{S}_d \subseteq \mathcal{S}$ $\mu$Services. At the same time, we consider the existence of $\mathcal{M} = \{1, 2, ..., M\}$ unit-demand users requesting exactly one $\mu$Service for the next period of allocation. Specifically, we denote with $\mathcal{M}_s$ the set of users requesting $\mu$Service $s \in \mathcal{S}$, which for unit-demand users means that $\mathcal{M}_s \cap \mathcal{M}_{s'} = \emptyset$ for any $s, s' \in \mathcal{S}$ where $s \neq s'$.

We assume that in each market, there is an Application Service Provider (AppSP) for a $\mu$Service $s$, aware of the latency between each user $m \in \mathcal{M}_s$ and device $d \in \mathcal{D}_s$ for the duration of their engagement. That is, based on the latency information, the AppSP derives the QoS $u_{md}$ produced by assigning user $m$ to device $d$. Furthermore, for each $\mu$Service $s$ there is an always available service computing infrastructure, *i.e.,* back-end Data Centre, providing a default QoS to a user $m \in \mathcal{M}_s$, $u_{m\emptyset}$, that given an expected latency the AppSP can estimate in a similar way.

### B. Multi-item Unit-Demand Auctions Preliminaries

Users requesting $\mu$Service $s \in \mathcal{S}$ are interested in achieving the highest possible QoS improvement, compared to the QoS provided by the back-end Data Centre, from their service by a device $d \in \mathcal{D}_s$. On the other hand, devices are interested in being compensated while maximising their revenue given their installed $\mu$Services. Clearly, the objectives of users and devices are conflicting with each other; that is, there would be difficulties in reaching an agreement regarding the requests served by each device, and the service price, if both parties were left alone to decide. Therefore, there is a need for a market mechanism that will ensure, as an independent entity, the effective assignment of user requests to $\mu$Service instances hosted by individual devices, *i.e.,* $x_s : \mathcal{M}_s \rightarrow \mathcal{D}_s \cup \{\emptyset\}$ $\forall s \in \mathcal{S}$ where $\{\emptyset\}$ corresponds to the default execution location of each $\mu$Service. Note that the back-end Data Centre is considered having sufficient capacity for serving the existing set of request as opposed to each device that has limited computing capabilities. In fact, for presentation purposes due to simplified notation, we assume that each device can serve up to a single user at each allocation period; that said, we show in Section III-C that the generalisation of the mechanism is straightforward.

In the context of a market, the assignment of requests to devices is determined by an auction mechanism. The corresponding auction falls into the category of *multi-item*, since there are multiple offered instances for a specific $\mu$Service, *unit-demand* auctions. In detail, the expected QoS improvement by each unit-demand user $m \in \mathcal{M}_s$ when served by a device $d \in \mathcal{D}_s$ corresponds to a monetary valuation:

$$v_{md} = f(u_{md}, u_{m\emptyset}) \tag{1}$$

where $f(\cdot, u_{m\emptyset})$ is an increasing function known by the AppSP that converts each user's QoS improvement to monetary values.[1]

Then given the available $\mu$Service $s \in \mathcal{S}$ instances, *i.e., supply*, at a given price vector $p_s = (p_{sd} : \forall d \in \mathcal{D}_s)$, where $p_{sd}$ $\forall d \in \mathcal{D}_s$ is the price of instance $s$ at device $d$, each user is aiming to maximise her net valuation, *i.e.,* valuation after price reduction $v_{md} - p_{sd}$. That is, user $m \in \mathcal{M}_s$ expresses

[1]Ideally, the $f(\cdot, u_{m\emptyset})$ function can capture the potential handovers of either the devices or the users, during the time-slot of interest, along with their impact on the $\mu$Service's QoS. However, we consider the network conditions between a user and a device stable at each time-slot since the prediction of mobility patterns is beyond the scope of this work.

her interest for those instances by her demand correspondence set:

$$D_m(p_s, r_s) = \{d \in \mathcal{D}_s : v_{md} - p_{sd} \geq v_{md'} - p_{sd'} \wedge \ p_{sd} \geq r_{sd}, \forall d' \in \mathcal{D}_s\} \quad (2)$$

where $r_s = (r_{sd} : \forall d \in \mathcal{D}_s)$ is the reservation price vector of offered $\mu$Service $s \in \mathcal{S}$ device instances. Specifically, we denote by $r_{sd}$ the reservation price at which device $d \in \mathcal{D}_s$ offers its instance $s$, meaning that the device will not serve any user requesting $\mu$Service $s$ for a lower price, *i.e.,* $p_{sd} < r_{sd}$ $\forall d \in \mathcal{D}_s$.

Given the demand correspondence set of each user, the market aims to derive an assignment $x_s : \mathcal{M}_s \rightarrow \mathcal{D}_s \cup \{\emptyset\}$ $\forall s \in \mathcal{S}$ with the following attributes:

- *A1*) each user is assigned uniquely to an instance from her demand correspondence or a default instance at the back-end Data Centre, *i.e.,* instance allocation $x_m \in D_m(p_s, r_s)$ for each $m \in \mathcal{M}_s$ where $x_{sm} \neq x_{sm'}$ for any $m' \in \mathcal{M} \backslash \{m\}$ or $x_{sm} = \emptyset$.
- *A2*) each device is either serving a single user for a price that is higher than its minimum possible (reservation) price, or up to a single user when its price is exactly equal to the minimum possible one, *i.e.,* if $x_{sm} = d$ for a user $m \in \mathcal{M}_s$, then $x_{sm} \neq x_{sm'}$ for any $m' \in \mathcal{M} \backslash \{m\}$ and $p_{sd} \geq r_{sd}$. Otherwise, if device's price is less than the minimum one then it does not serve any user, *i.e.,* if $p_{sd} < r_{sd}$ then $x_{sm} \neq d$ for any user $m \in \mathcal{M}_s$.

Under conditions *A1*) and *A2*) the market is reaching an *equilibrium state* that is characterised by an *equilibrium assignment* $x_s^*$, where users are satisfied with their assigned instance, and an *equilibrium price vector* $p_s^*$ $\forall s \in \mathcal{S}$. In fact, it has been shown by Shapley [31] that the set of equilibrium vectors is non-empty while forming a complete lattice that guarantees the existence of a unique minimal equilibrium vector, given the reservation price of each instance. In literature, this unique vector is referred as Vickrey-Clarke-Groves (VCG) equilibrium assignment, $x^{\text{VCG}}$, associated with the VCG equilibrium price vector, $p^{\text{VCG}}$ [38]. An auction mechanism that achieves the VCG equilibrium guarantees to each user that her assigned instance is offered to *the minimum possible price over all possible equilibrium price vectors*, *i.e.,* the user has no incentives in acquiring the instance in any other equilibrium price, which is a highly desirable property. The reason is that in a market where instances are assigned according to the VCG equilibrium, end users are truthful with respect to their valuations about a $\mu$Service instance, *i.e.,* each user $m \in \mathcal{M}_s$ is not incentivised to submit an untruthful valuation, *i.e.,* $< v_{md}$, about an instance at device $d \in \mathcal{D}_s$.

## III. DEEM Mechanism

In this section we present DEEM's components, namely:

1) DEEM Market Operation: That is the auction mechanism applied to each application specific market.
2) DEEM Orchestrator: That allocates each device to the request that values it the most, by updating the reservation price of each device to a market iteratively.

For the simplicity of the mechanism presentation we assume that each application provider operates a single market, while each device serves at most a single user; however, the mechanism can be easily generalised as we discuss in Section III-C.

### A. DEEM Market Operation

We describe DEEM market operation component by focusing on a single $\mu$Service market $s \in \mathcal{S}$. The presented component relies on Vickrey-English-Dutch (VED) auctions that have the attribute of deriving the VCG equilibrium assignment along with the corresponding price vector. This component for a $\mu$Service $s$ considers as given the set of $\mathcal{D}_s$ devices, devices' price vector $p_s$, reservation price vector $r_s$, and users $\mathcal{M}_s$ valuations. Then each device of the market is categorised in one of the following sets:

- *Universally allocated devices*, $\tilde{\mathcal{D}}_s(p_s, r_s)$, defined as the set of devices that satisfy at their current price at least 2 users, *i.e.,* $\exists m, m' \in \mathcal{M}_s : d \in D_m(p_s, r_s) \cap D_{m'}(p_s, r_s)$ such that $m \neq m'$ $\forall d \in \tilde{\mathcal{D}}_s(p_s, r_s)$, while even by removing a user the set of allocated items is not depleted, *i.e.,* after excluding randomly a user all devices in $\tilde{\mathcal{D}}_s(p_s, r_s)$ can still be assigned. The set of universally allocated devices can be identified in polynomial time by the FindUnivAllocItems procedure presented in [29].
- *Withheld devices*, $\mathcal{WD}_s(p_s, r_s)$, including the devices whose price is less than their reservation price or equal to their reservation price without being universally allocated, *i.e.,* $\mathcal{WD}_s(p_s, r_s) = \{d \in \mathcal{D}_s \backslash \{\tilde{\mathcal{D}}_s(p_s, r_s) : p_{sd} \leq r_{sd}\}$. These devices are not assigned to any user. Note that the derivation of $\mathcal{WD}_s(p_s, r_s)$ set of devices is trivial given the current vector price $p_s$ and set $\tilde{\mathcal{D}}_s(p_s, r_s)$.
- *Devices in excess supply*, $\mathcal{ES}_s(p_s, r_s)$, that consists of the devices that neither belong to the universally allocated nor the withheld devices, *i.e.,* $\mathcal{ES}_s(p_s, r_s) = \{\mathcal{D}_s\} \backslash \{\tilde{\mathcal{D}}_s(p_s, r_s) \cup \mathcal{WD}_s(p_s, r_s)\}$. That is, $\mathcal{ES}_s(p_s, r_s)$ can be also identified in polynomial time since the main overhead is the identification of $\tilde{\mathcal{D}}_s(p_s, r_s)$ set which is polynomial.
- *Devices in excess demand*, $\mathcal{ED}_s(p_s, r_s)$, which is a subset of the universally allocated devices, $\mathcal{ED}_s(p_s, r_s) \subseteq \tilde{\mathcal{D}}_s(p_s, r_s)$, where *i*) the number of devices in each proper subset $T \subset \mathcal{ED}_s(p_s, r_s)$, is strictly smaller than the number of users that demand a device in $T$, and *ii*) the users that demand at least a single device in $\mathcal{ED}_s(p_s, r_s)$ they do not request devices outside the $\mathcal{ED}_s(p_s, r_s)$ set. Authors in [11] prove that for a given price vector there exists a *unique set in excess demand with maximal cardinality*, $\mathcal{ED}_s^*(p_s, r_s)$, that can be identified in polynomial time by applying the "Ford and Fulkerson" algorithm [19].

It is easy to see that the aforementioned sets cover all the potential states that a device instance can belong, given the price and reservation price vectors, $p_s$ and $r_s$ respectively. In particular, following the depicted devices in Fig. 2 we see that if the price of a device $d$ is lower than its reservation price, $p_{sd} < r_{sd}$, or exactly equal to its reservation price

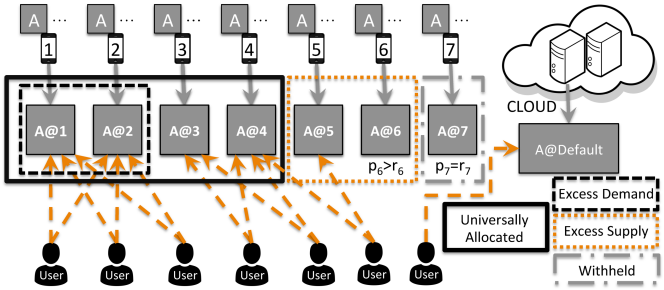



Fig. 2: Edge $\mu$Service-Specific Market, $\mu$Service A instance

without being universally allocated then the device belongs to the set of withheld devices, $d \in \mathcal{WD}_s(p_s, r_s)$, like device 7. Otherwise, if device's price is higher that its reservation one and it is requested by at most a single user it belongs to devices in excess supply, $d \in \mathcal{ES}_s(p_s, r_s)$, such as devices 5 and 6. In case that each device is requested by at least 2 users and any device can continue being allocated even by excluding a user from the auction, then it belongs to the set of universally allocated devices, $d \in \tilde{\mathcal{D}}_s(p_s, r_s)$, as the devices 1, 2, 3, and 4. The excess demand devices is a subset of the universally allocated ones that cannot satisfy the demand correspondence of their involved users, like devices 1 and 2 that are requested by 3 users. Note that even if we exclude a single user out of the participating ones, the universally allocated devices 1 to 4 would still be able to be assigned to a user. Lastly, we consider an always available $\mu$Service Data Centre that serves all the requests that fail to be assigned to a device's instance.

DEEM's market operation component derives the VCG equilibrium of the market, given the reservation prices vector, by applying a VED auction mechanism. In VED auctions, the price is adjusted in a way that eliminates the devices in excess supply, $\mathcal{ES}_s(p_s, r_s)$, as well as in excess demand, $\mathcal{ED}_s(p_s, r_s)$, since both sets are associated to the VCG equilibrium state by:

**Theorem 1:** A price vector is the VCG equilibrium price vector of the market if and only if the sets of devices in excess demand and supply are empty, *i.e.,* $p_s = p_s^{\text{VCG}}$ iff $\mathcal{ES}_s(p_s, r_s) = \mathcal{ED}_s(p_s, r_s) = \{\emptyset\}$.

The theorem is proved in [25], leading us to the following straightforward corollary:

**Corollary 1:** Given the reservation price of device $d$ at market $s \in \mathcal{S}_d$, $r_{sd}$, if the market is at a VCG equilibrium state then $d$ will either belongs to the set of universally allocated devices, $d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG}}, r_s)$, at a price $p_{sd}^{\text{VCG}} \geq r_{sd}$, or at the set of withheld devices, $d \in \mathcal{WD}_s(p_s^{\text{VCG}}, r_s)$, at a price $p_{sd}^{\text{VCG}} \leq r_{sd}$.

That is, VED auctions iteratively attempt to eliminate the sets of excess demand and supply by adjusting the price of a device $d \in \mathcal{D}_s$ at the $k$-iteration, $p_{sd}^k$, according to:

$$p_{sd}^k = \begin{cases} p_{sd}^{k-1} + \Delta p, & \text{if } d \in \mathcal{ED}_s^*(p_s^{k-1}, r_s), \\ p_{sd}^{k-1} - \Delta p, & \text{if } d \in \mathcal{ES}_s(p_s^{k-1}, r_s), \\ p_{sd}^{k-1} & \text{otherwise.} \end{cases}$$

where if $d \in \mathcal{ED}_s^*(p_s^{k-1}, r_s)$ the price is increased by $\Delta p$, *i.e.,* eliminating the excess demand set upon iteration $k-1$; on the other hand, if $d \in \mathcal{ES}_s(p_s^{k-1}, r_s)$ the price is decreased by $\Delta p$, *i.e.,* eliminating the excess supply set upon iteration $k-1$. Note that the value of $\Delta p$, is considered small enough to capture the differences between users' valuations, *i.e.,* $\Delta p \leq |u_{md} - u_{m'd'}|$ for any $m, m' \in \mathcal{M}_s$ and $d, d' \in \mathcal{D}_s$ where $m \neq m'$ or $d \neq d'$. We refer to the policy of increasing the prices of each VM in $\mathcal{ED}_s(p_s, r_s)$ as *$\mathcal{ED}$-increase* and to the policy of decreasing the prices in $\mathcal{ES}_s(p_s, r_s)$ as *$\mathcal{ES}$-decrease*. The next lemmas indicate that *VED auctions can derive the VCG equilibrium of a $\mu$Service specific market in polynomial time*, *i.e.,* the excess demand and supply sets are eliminated in polynomial time as proved in [35].

**Lemma 1:** Consecutive applications of *$\mathcal{ED}$-increase* (*$\mathcal{ES}$-decrease*) policy eliminate the excess demand $\mathcal{ED}_s(p_s, r_s)$ (excess supply $\mathcal{ES}_s(p_s, r_s)$) set of a $\mu$Service specific $s$ market in polynomial time.

However, applying both *$\mathcal{ED}$-increase* and *$\mathcal{ES}$-decrease* policies at the same iteration might trap the process into a cycle, *i.e.,* the set of excess supply and demand return to their previous state after a number of iterations as it is shown in [24]. Therefore, the *$\mathcal{ED}$-increase* (*$\mathcal{ES}$-decrease*) policy has to be applied in isolation in each iteration until completely eliminating $\mathcal{ED}_s(p_s, r_s)$ ($\mathcal{ES}_s(p_s, r_s)$) before changing to policy *$\mathcal{ES}$-decrease* (*$\mathcal{ED}$-increase*) targeting the set $\mathcal{ES}_s(p_s, r_s)$ ($\mathcal{ED}_s(p_s, r_s)$). The convergence to a price vector where both sets of excess supply and excess demand are empty, $\mathcal{ED}_s(p_s, r_s) = \mathcal{ES}_s(p_s, r_s) = \{\emptyset\}$, is guaranteed by the following monotonicity lemma proved in [12]:

**Lemma 2:** For any price vector $p_s^k \geq 0$ and reservation price vector $r_s$, i) If $\mathcal{ES}_s(p_s^k, r_s) = \{\emptyset\}$ and an *$\mathcal{ED}$-increase* price adjustment policy is applied at iteration $k$, then $\mathcal{ES}_s(p_s^{k+1}, r_s) = \{\emptyset\}$; similarly, ii) if $\mathcal{ED}_s(p_s^k, r_s) = \{\emptyset\}$ and an *$\mathcal{ES}$-decrease* price adjustment policy is applied at iteration $k$, then $\mathcal{ED}_s(p_s^{k+1}, r_s) = \{\emptyset\}$.

### *B. DEEM Orchestrator*

DEEM's orchestrator component, is responsible for associating uniquely a device to the $\mu$Service market that maximises its profit[2]. Clearly, each device has a preference in serving the $\mu$Service request that is willing to pay the most. DEEM controls devices' participation into each market by adjusting their reservation prices iteratively. The operation of DEEM's orchestrator follows the VCG equilibrium state derivation at each market, that provides information related to $\mu$Service prices that a device can be assigned to.

In detail, consider that DEEM's market operation components has derived the VCG equilibrium state of each market for the $j$-th reservation price update time. Let $r_d^j = (r_{sd}^j : s \in \mathcal{S}_d)$ and $p_d^{\text{VCG},j} = (p_{sd}^{\text{VCG},j} : s \in \mathcal{S}_d)$ be the reservation price and the VCG equilibrium price vector of device $d$ over its participating markets. From Corollary 1 we know that if a universally allocated device $d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j}, r_s^j)$ increases its reservation price by $r_{sd}^{j+1} > p_{sd}^{\text{VCG},j}$, it will either remain universally allocated at a higher price in the next iteration $j+1$, *i.e.,* $p_{sd}^{\text{VCG},j+1} \geq r_{sd}^{j+1} > p_{sd}^{\text{VCG},j}$, or it will move to the

[2]Under the assumption that each device can serve up to a single user.

set of withheld devices. Although the assignment of device $d$ at the increased reservation price is uncertain, such an action certainly increases the price of *other* devices in an $\mu$Service $s$ market under the conditions expressed next:

**Lemma 3:** Let $d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j}, r_s^j)$ at price $p_{sd}^{\text{VCG},j}$ after iteration $j$ and $r_{sd}^{j+1} > p_{sd}^{\text{VCG},j}$ be its updated reservation price for iteration $j+1$. Then if the demand correspondence of its assigned user $m \in \mathcal{M}_s$, $D_m(p_s^{\text{VCG},j}, r_s^j)$, its not a singleton, and $\forall d' \in \mathcal{D}_s \backslash \{d\}$ $r_{sd'}^{j+1} = r_{sd'}^j$, in iteration $j+1$ the VCG price of at least another device in the market will increase.

*Proof.* Let market $s$ to initiate its next iteration execution from its previously found equilibrium prices, *i.e.,* $p_s^{1,j+1} = p_s^{\text{VCG},j}$, and device $d$ be the only one with an updated reservation price $r_{sd}^{j+1} > p_{sd}^{\text{VCG},j}$. Then in the beginning of iteration $j+1$, device $d$ will move immediately to the set of withheld devices, since $p_{sd}^{1,j+1} = p_{sd}^{\text{VCG},j} < r_{sd}^{j+1}$, and will be also excluded from the demand correspondence of user $m$, *i.e.,* $D_m(p_s^{1,j+1}, r_s^{j+1}) = D_m(p_s^{\text{VCG},j}, r_s^j) \backslash \{d\}$, as indicated by Eq. (2). Since $D_m(p_s^{\text{VCG},j}, r_s^j)$ is not a singleton, there is *at least* a single device $d' \in D_m(p_s^{1,j+1}, r_s^{j+1})$ that is previously assigned, in iteration $j$, to a user $m' \in \mathcal{M}_s \backslash \{m\}$. Hence, $d'$ is requested by an additional user, $m$, meaning that the excess demand set is not empty anymore, $d' \in \mathcal{ED}_s^*(p_s^{1,j+1}, r_s^{j+1}) \neq \{\emptyset\}$. Therefore the $\mathcal{ED}$-increase policy will be called to increase the price of device $d'$. Lastly, the excess supply set will remain empty due to monotonicity Lemma 2, and therefore there will be no price reductions. □

Note that even if the demand correspondence of the assigned user was a singleton, the price of another device might be increased again but there is no guarantee about it. Furthermore, if the reservation prices in each iteration can only increase, we have that:

**Lemma 4:** Let $d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j}, r_s^j)$ at price $p_{sd}^{\text{VCG},j}$ after iteration $j$ and $r_{sd}^{j+1} = r_{sd}^j$ while $r_{sd'}^{j+1} > r_{sd'}^j$ $\forall d' \in \mathcal{D}_s \backslash \{d\}$. Then $d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j+1}, r_s^{j+1})$ at a price $p_{sd}^{\text{VCG},j+1} \geq p_{sd}^{\text{VCG},j}$ in iteration $j+1$.

*Proof.* Let market $s$ to initiate its next iteration execution from its previously found equilibrium prices, *i.e.,* $p_s^{1,j+1} = p_s^{\text{VCG},j}$, with a reservation price for device $d$ $r_{sd}^{j+1} = r_{sd}^j$ while for the rest of devices $\forall d' \in \mathcal{D}_s$ $r_{sd'}^{j+1} > r_{sd'}^j$. Then in the beginning of iteration $j+1$, device $d$ will remain to the set of universally allocated devices at price $p_{sd}^{\text{VCG},j+1} = p_{sd}^{\text{VCG},j}$ since $d$ is still included in the demand correspondence of the users that requested it. However, a reservation price increase to the rest of devices, as we see from Lemma 3, can move $d$ to the set of excess demand devices which price will increase by the $\mathcal{ED}$-increase policy, resulting in a $p_{sd}^{\text{VCG},j+1} > p_{sd}^{\text{VCG},j}$. Note that again the excess supply set will remain empty due to monotonicity Lemma 2, and therefore there will be no price reductions. □

In summary, from Lemma 4 we have that a universally allocated device can either retain its reservation price that guarantees a VCG equilibrium price at least equal to the current one, while from Lemma 3 we see that by increasing the reservation price to a market the device either achieves a higher price or it is not assigned to a user. DEEM orchestrator component leverages these facts to identify the most profitable market for each device by *i*) retaining the reservation price to the market that achieves the highest price, while *ii*) increasing the reservation price to the rest of markets in an effort to either achieve an even higher price or exclude this market.

Therefore for a device $d$, DEEM's orchestrator component identifies the most profitable market at iteration $j$:

$$s_d^{\text{max},j} = \arg\max_{s \in \mathcal{S}_d} \{p_{sd}^{\text{VCG},j} : d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j}, r_s^j)\}. \quad (3)$$

at price

$$p_d^{\text{max.},j} = \arg\max_{s \in \mathcal{S}_d} \{p_{sd}^{\text{VCG},j} : d \in \tilde{\mathcal{D}}_s(p_s^{\text{VCG},j}, r_s^j)\}. \quad (4)$$

and updates the reservation prices of $d$ for the next iteration $j+1$ according to the *reservation-price-update policy*:

$$r_{sd}^{j+1} = \begin{cases} r_{sd}^j, & \text{if } s == s_d^{\text{max},j}, \\ p_d^{\text{max.},j} + \Delta p, & \text{otherwise.} \end{cases}$$

Note that in case than more than a single market achieves the maximum price, DEEM orchestrator selects one out of them randomly.

**Theorem 2** DEEM mechanism assigns uniquely a device to its most profitable market in polynomial time.

*Proof.* Clearly, in each orchestrated iteration $j+1$ the *reservation-price-update policy* will either increase the maximum price for device $d$, $p_d^{\text{max.},j+1} > p_d^{\text{max.},j}$, or it will retain price $p_d^{\text{max.},j+1} = p_d^{\text{max.},j}$ in market $s_d^{\text{max},j}$ while moving the device to the set of withheld devices in any other market $s' \in \mathcal{S}_d \backslash \{s_d^{\text{max},j}\}$. Therefore the device will be considered associated uniquely to its most profitable market. The minimum price increase is $\Delta p$, and therefore the price cannot continue increasing for more than $\lceil \bar{p}/\Delta p \rceil$ times, where $\bar{p}$ is the maximum possible valuation in any auction defined as $\bar{p} = \max_{\forall m \in \mathcal{M}_s, d \in \mathcal{D}_s} \{v_{md}\}$. Since DEEM's market operation component is executed in polynomial time, as we know from Lemma 1, multiplying its time with the maximum number of price increase from the reservation-price-update policy, $\lceil \bar{p}/\Delta p \rceil$, will retain DEEM's execution time polynomial. Therefore, DEEM mechanism assigns uniquely a device to its most profitable market in polynomial time. □

### C. DEEM Extensions

*1) Multiple Users Service Capabilities Devices:* Our model and mechanism description has considered so far the scenario where each device can serve up to a single user request. However, this is not the general case where more powerful computing equipment, like the one deployed over electric cars, can support multiple user requests at the same time. DEEM can be adapted to this setting by decomposing each computing equipment to identical devices that can handle up to a single user request.

*2) Multiple Markets per Application Service Provider:* Up to this point, we assumed that each application provider operates a single market. However, the presented mechanism can scale in any number of application specific markets. In detail, devices can participate, when it is required, in these markets simultaneously as they do in the case of multiple applications. On the other hand, user requests have again to choose a single application specific market to participate. The user to market association problem is considered orthogonal to this work.

## IV. Performance Evaluation

In this section, we demonstrate the performance of DEEM for various system parameters and compare it with different approaches. At first, we describe the evaluation setting of our experiments before presenting the corresponding results.

### A. Evaluation Setting

*1) Mobility Traces and Cellular infrastructure:* We evaluate DEEM in an urban setting by leveraging the mobility dataset [3] generated in the context of TAPASCologne project [37] using a custom-built, Python-based simulator. The dataset consists of 700,000 individual vehicle journeys that take place over 24 hours in an area of 400 square kilometres. We consider each vehicle as a mobile user, associated with a mobile device, while focusing on an off-peak hour of the dataset that presents a good ratio of average vehicle speed, *i.e.,* 30 km/h, and number of available vehicles at each second, *i.e.,* 5200 vehicles. The number of vehicles remains approximately constant throughout the duration of the simulation, with on average 13 vehicles leaving/arriving at every second. The simulation area is divided into 864 hexagon cells of 1 km radius. Vehicles are connected to a single cell at each time, while their in-between communication is taking place via their corresponding base station. We consider a tree-like backhaul topology [26] with an inter-cell latency equal to 10 ms; that is, the latency between cells increases linearly as a function of the number of cells that separate them. Lastly, we assume that a vehicle on average requires approximately 5 ms to access the base station that it is currently connected to [4], *e.g.,* the latency between 2 vehicles that are connected on different cells that are located 3 cells away is 40 ms decomposed into the 30 ms of backhaul latency and the latency of 10 ms required by both vehicles to access their base station.

*2) QoS of Engagement with $\mu$Services:* Naturally, the QoS for an end user engaging with a $\mu$Service is a decreasing function of user's perceived latency to the device hosting the $\mu$Service in terms of RTT [17]. In particular, we make the assumption that the QoS decreases linearly with increasing RTT latencies as depicted in Fig. 3. In this figure, the constant $u_{\max}$ ($u_{\min}$) represents the maximum (minimum) QoS that the $\mu$Service user can achieve at the minimum (maximum)

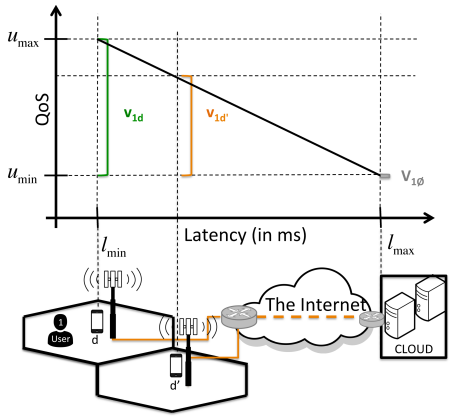



Fig. 3: $\mu$Services QoS Function of Latency

latency $l_{\min}$ ($l_{\max}$), *i.e.,* $u(l_{\min}) = u_{\max}$ and $u(l_{\max}) = u_{\min}$. We set $u_{\max} = 100$, $l_{\min} = 10$ ms, and $l_{\max} = 100$ ms for each $\mu$Service type assuming that all devices have identical latencies to their back-end Data Centre, *i.e.,* cloud, locations[5]. We set the number of $\mu$Service types equal to 1000, while each $\mu$Service type in our system is assigned randomly a $u_{\min}$ value in the interval of 0 to 100 at the granularity of a single decimal digit. Therefore, the QoS gains for a $\mu$Services varies from 0.0, for $u_{\min} = 100.0$, to 100.0 for $u_{\min} = 0.0$. Note that the auction's price adjustment increment is set to $\Delta p = 0.1$ in order to capture the granularity of QoS gains.

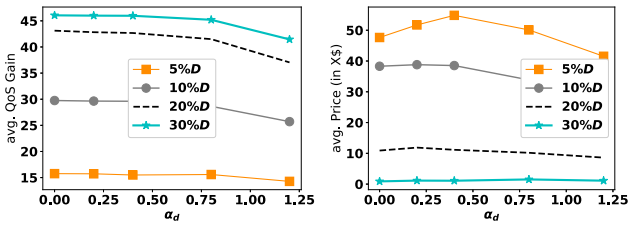
*3) Participation of Users and Devices:* Given the statistics of surveys [4] we assume that the set of users, actively demanding an $\mu$Service at a random moment, consists of approximately the 20% of 5200 vehicles, *i.e.,* on average 1000 users are actively engaged to a $\mu$Service at anytime. Furthermore, by default 20% of the 5200 vehicles constitute devices that participate to lease $\mu$Service instances. We consider that upon engaging to a device, users do not submit new requests; that defines an one-to-one user-device association. Moreover, each participating device is able to process at most a single user request at any given time no matter how many $\mu$Services might support (see below for device $\mu$Service deployment settings). We evaluate the impact of device participation on the performance in Section IV-B1.

*4) Deployment of $\mu$Services and Request Generation:* The engagement duration of each user to a $\mu$Service follows an exponential distribution which dictates an average engagement duration of 1 minute. The number of $\mu$Services supported at each device is determined according to the statistics provided in [5] where i) the 6% has no services, ii) the 32% has installed 1 to 10 $\mu$Services, iii) the 33% 10 to 20, iv) the 17% 21 to 30, and v) finally the remaining 12% 31 to 40 $\mu$Services. The specific set of $\mu$Service types supported at each device is determined by a Zipf distribution of exponent $\alpha_d$ that favours the most QoS sensitive $\mu$Services, *i.e.,* the most popular services are the ones with the highest QoS gain at a given latency $x$, $u(x) - u_{\min}$. In that way, we capture devices'

[3] Available at http://kolntrace.project.citi-lab.fr

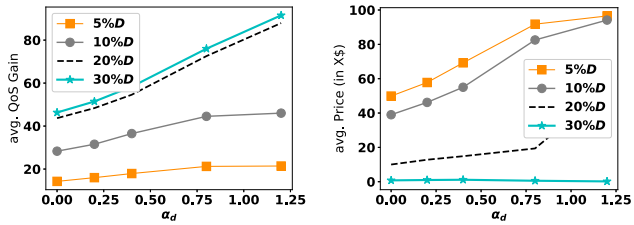
[4] With recent advances in LTE technology, mobile operators reported handset-to-base-station latencies around 2 ms (RTT of 4 ms), see: http://news.itu.int/with-5g-looming-sk-telecom-reduces-lte-latency-to-just-2ms

[5] 100 ms is the average latency observed for Amazon Web Services according to CloudPing, available at http://www.cloudping.info.




(a) Avg. QoS Gain (b) Avg. Device Price

Fig. 4: Scenario 1: Uniform $\mu$Service requests for varying deployment distribution.




(a) Avg. QoS Gain (b) Avg. Device Price

Fig. 5: Scenario 2: Varying $\mu$Service requests for varying deployment distribution.

need to support the $\mu$Service types that achieve the highest possible QoS when served locally as opposed to being served at a distant Data Centre. Similarly, $\mu$Services are requested by the users according to another Zipf distribution of exponent $\alpha_r$ that favours the most QoS sensitive $\mu$Services as well. Lastly, we execute DEEM mechanism every 5 seconds.

### B. Simulation Results

We evaluate the performance of the system in terms of *i)* average QoS Gain and *ii)* average derived prices by the auction process. In Section IV-B1, we investigate the impact of both $\mu$Services' deployment and request distributions on DEEM's performance for different device participation levels. Then in Section IV-B2, we compare DEEM against edge computing, greedy and First Come First Served user-device association approaches.

*1) Impact of $\mu$Service Demand and Deployment Distributions:* We investigate the impact of $\mu$Service deployment and demand distribution on average QoS and prices by considering two scenarios. In both scenarios, the $\mu$Service deployment exponent, $\alpha_d$, takes values in the interval of 0 to 1.2. Then in the first scenario, user requests follow a uniform distribution, *i.e.,* $\alpha_r = 0.0$, while in the second requests popularity distribution is identical to the microservices' deployment distribution, *i.e.,* $\alpha_r = \alpha_d$. We consider a linear association between the average QoS gain and the price of instances, *i.e.,* a unit of QoS gain is estimated in X dollars.

We evaluate DEEM under different device participation percentages that offer their $\mu$Service instances, namely 5%, 10%, 20%, and 30%. Since DEEM provides an one-to-one user-to-device association while the users requesting a $\mu$Service account for the 20% of devices, the participation percentages capture the settings of $\mu$Service instance *i*) under-provisioning, for the service device participations of 5% and 10%, *ii*) ideal provisioning, for a device participation equal to users demand, *i.e.,* 20% participation, and *iii*) over-provisioning, for the case of 30% service device participation.

**Scenario 1:** In Fig. 4, we present the results for the first scenario of uniform requests, $\alpha_r = 0$. To begin with, in Fig. 4a, we demonstrate the QoS gains of DEEM at different participation percentages of devices, *i.e.,* each falling under one of the three provisioning settings, for a varying deployment of supported $\mu$Service types at devices, according to the Zipf exponent $\alpha_d$. We observe that increasing $\alpha_d$ eventually leads to reduced QoS gains for all participation percentages as a consequence of having a skewed distribution of available $\mu$Services in devices while requesting $\mu$Services uniformly. As expected, the highest QoS gains are achieved in the over-provisioning scenario while the lowest in the under-provisioning since more requests are directed to a Data Centre having a QoS gain equal to 0.

In Fig. 4b, we demonstrate how DEEM derives prices for the same scenario. Unlike the average QoS gain metric which takes users that are assigned to a Data Centre into account, the average price metric does not include those users in the price computation. As a result, we observe the highest average price for the under-provisioning scenario, *i.e.,* 5% participation, where many users compete for a limited number of $\mu$Service instances. At the same time, both the ideal- and over-provisioned scenarios achieve lower average prices than their QoS gains. This is due to the competition factor of DEEM's price derivation process. Finally, we observe that average prices are reduced for the ideal- and under-provisioned cases.

**Scenario 2:** We present the scenario for the Zipf-distributed $\mu$Service request popularity in Fig. 5, where $\alpha_r$ is set to be equal to $\alpha_d$. In this case, both the QoS gain and prices increase function of $\alpha_d$ as shown in Figs. 5a and 5b respectively. This increase is expected since requests popularity and $\mu$Service availability are aligned. Note that as in the previous scenario, the average price remains low at a higher instance provisioning settings.

*2) Impact of Design Choices:* DEEM's design is based on the choices of *i*) $\mu$Service-specific, as well as *ii*) VED auction based markets. Both choices come with the advantage of reducing the execution time of such a mechanism. Clearly, the smaller a market is, in terms of users and participating devices, the faster the assignment of requests to microservice instances becomes. That is, $\mu$Services' request and deployment distribution is affecting DEEM's execution time, as we see in Table I [6]. Specifically, DEEM's execution time is prone to the size of the *largest* microservice market. That is, for the case of uniform deployment and demand, *i.e.,* $\alpha_r = 0.0$,

[6]The execution times are computed by using a 2.2 GHz Intel Core i5 processor.

TABLE I: EXECUTION TIME, DEEM VED VS. DEEM ENGLISH VS. CENTRALISED VED

| Setting | DEEM | DEEM English | Centralised VED |
|---|---|---|---|
| $\alpha_r = 0.0$, $\alpha_d = 0.0$ | 0.795s | 0.955s | 198.020s |
| $\alpha_r = 0.0$, $\alpha_d = 0.8$ | 0.810s | 0.993s | 197.814s |
| $\alpha_r = 0.8$, $\alpha_d = 0.8$ | 5.207s | 5.706s | 198.122s |

$\alpha_d = 0.0$, where the markets are on average of equal size, we have the minimum possible execution time for our setting, *i.e.,* lower than 0.8 seconds as we see in the second column of Table I. DEEM's performance is deteriorated negligibly when increasing the number of participating devices in some markets, by augmenting the deployment exponent $\alpha_d$ to 0.8, with an execution time that just exceeds the 0.8 seconds. The worst execution time for DEEM is achieved when some microservice markets are becoming significantly popular compared to the others, *i.e.,* $\alpha_r = 0.8$, $\alpha_d = 0.8$, where the execution duration exceeds the 5 seconds.

Clearly, for the case of centralised markets where all devices and all users are participating at a single market, the time is not affected by the deployment and request distribution of microservices although the execution time is bigger by 3 orders of magnitude compared to the $\mu$Service-specific execution. Lastly, DEEM by leveraging the previously found market prices of the market, via the application of VED auctions, achieve a reduction of 10%, as opposed to English auctions [29] where in each execution every device has an initial price equal to 0. The advantages of exploiting VED initial price attributes, in a context where auctions are applied periodically, have been also presented in [35].

*3) Comparison:* Here we compare DEEM under $\mu$Service requests exponent $\alpha_r = 0.3$ against the following strategies:

1) *Greedy:* This is a peer-to-peer (P2P) approach which pairs the requests for $\mu$Services by prioritising the users with the highest valuations first. Specifically, after collecting all user requests for $\mu$Service types and device offers for $\mu$Service instances, the greedy approach sorts all possible user-device pairs based on their matching $\mu$Service types and then goes through the pairs assigning users to devices, if they are both unassigned so far, starting from the highest user-to-device valuations.
2) *First Come First Served (FCFS):* The FCFS is also a P2P approach which serves $\mu$Service requests in the order that they are received. That is, users and devices are paired according to the arrival order of the requests for $\mu$Service and $\mu$Service instance offers.
3) *Edge-MAP:* This is an MEC approach with cloudlets deployed at the base stations described in [35].

Edge-MAP offers the optimal baseline where cloudlets provide their resources in the form of virtual machines capable of instantiating *any* $\mu$Service type in an on-demand fashion, *i.e.,* as requests arrive. Therefore Edge-MAP is not subject to the availability of specific $\mu$Service instances on devices as in the case of DEEM, Greedy, and FCFS settings. In Edge-MAP we consider a deployment of 50 edge cloudlets each capable of supporting up to 20 virtual machines at any time. This accounts for 1000 virtual machines that can serve 1000 user requests at-a-time. For a fair comparison, we assigned DEEM, FCFS, and Greedy strategies with the equivalent amount of participation rate (*i.e.,* 20%) that creates a pool of 1000 $\mu$Service instances, albeit tied to a specific service type unlike the cloudlet virtual machine resources.

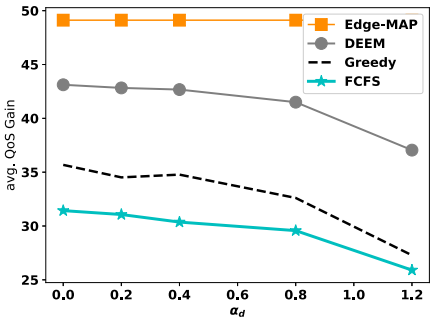



Fig. 6: Edge-MAP vs. DEEM vs. Greedy vs. FCFS, 20% devices participation, $\alpha_r = 0.3$.

As shown in Fig. 6, DEEM achieves significantly higher QoS gains than the Greedy and FCFS approaches. Furthermore, DEEM implements a truthful pricing scheme, where users have no incentives of misrepresenting their valuations for $\mu$Service instances in order to obtain financial gains as a consequence of VCG equilibrium assignment. On the other hand, Greedy and FCFS approaches are prone to untruthful valuations, which we ignore in their performance evaluation.

Edge-MAP scenario represents the MEC deployment at the edges. We observe that DEEM achieves slightly lower QoS gains than the Edge-MAP. However, it is important to note that Edge-MAP is not affected by $\alpha_d$, unlike DEEM, Greedy and FCFS strategies. This is because Edge-MAP can instantiate on-demand any $\mu$Service type in a virtual machine based on the actual demand and is not bounded to the limitations of $\mu$Service availability installed on devices. As a result, Edge-MAP achieves a constant QoS gain as shown in Fig. 6. Note however, that this advantage comes at the cost of deploying edge computing resources at base stations, while DEEM achieves similar benefits by exploiting existing devices.

## V. RELATED WORK

Several industrial platforms have been launched aiming to realise a vision of global decentralised computing that automatically rewards participating workers. Those platforms allow offloading video rendering [8] and data analytic [9] tasks, develop a cloud-like services platform based on fog computing as backend [6], [7] or perform a broad range of different computational tasks [10], [13]. Although all of the above-mentioned projects move drastically towards decentralisation, they neither provide mechanisms to automatically match workers with requests nor determine a concrete service pricing scheme. In this respect, DEEM is a significant con-

tribution to a critical system component that will make those platforms usable in real-world scenarios.

Limited efforts in the context of Fog/Edge Computing [16] consider the delivery of applications with low latency requirements by bringing application instances *closer* to their end users. In [21] authors propose a service/application provisioning using a heavy-weight combinatorial auction mechanism where application providers bid periodically for a number of virtual machines located at different cloudlet locations. More practical approaches are presented for edge computing settings [35] and in-network on-path provisioning mechanisms [14], [34], [36] where user requests allocate virtual machines on-demand for instantiating their requested applications. That said, unlike DEEM these approaches rely on the existence of a computing infrastructure which requires significant capital investments. Closer to our work is a mobile app sharing approach proposed in [27]; however, this work only describes a high-level solution without a mechanism to pair user requests to executing apps in other users' devices.

## VI. Conclusions

Nowadays, it is clear that native apps can benefit from a web-based, download/installation-free access approaches in order to overcome discoverability challenges. Web-based approaches can leverage decentralised computational solutions as well as microservice architectures in order to deliver native apps at an improved Quality of Service (QoS). In this paper, we proposed DEEM, a framework that relies on microservice-specific markets, acting as rendezvous points between user requests and available device microservice instances, to deliver applications in a P2P manner. In DEEM, markets allocate instances to the users that value them the most, in terms of QoS gains, while maximising devices income by associating them with the most profitable markets. Our simulation results demonstrate DEEM's ability in exploiting the pool of device instances for improving the QoS of applications with low latency requirements.

P2P sharing of computing resources is a promising research direction, which unlike mobile edge computing (MEC), does not require any infrastructure investment and it is self-scalable. Our simulation results demonstrate that DEEM achieves comparable performance to MEC approaches at the cost of a lightweight auction computation.

## Acknowledgment

This work has been supported by the EC H2020 ICN2020 project (GA no. 723014) and the EPSRC INSP fellowship (EP/M003787/1).

## References

[1] App download and usage statistics (2018): http://www.businessofapps.com/data/app-statistics/.
[2] Number of apps available in leading app stores as of 3rd quarter 2018: https://www.statista.com/statistics/276623/number-of-apps-available-in-leading-app-stores/.
[3] Number of mobile app downloads worldwide in 2017, 2018 and 2022: https://www.statista.com/statistics/271644/worldwide-free-and-paid-mobile-app-store-downloads/.
[4] Smart insights survey: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics.
[5] Statista insights applications decployment: https://www.statista.com/statistics/267309/number-of-apps-on-mobile-phones/.
[6] Dadi documentation. https://icosbull.com/eng/ico/dadi/whitepaper, 2018.
[7] Decentralized fog computing platform. https://sonm.com/, December 2018.
[8] Golem network. https://golem.network/, December 2018.
[9] Hypernet.io. https://hypernetwork.io/HypernetWhitePaper_v1.1.pdf, 2018.
[10] iexec whitepaper. https://iex.ec/whitepaper/iExec-WPv3.0-English.pdf, 2018.
[11] T. Andersson, C. Andersson, and A. Talman. Sets in excess demand in simple ascending auctions with unit-demand bidders. *In Springer Annals of Operations Research, 2013*.
[12] T. Andersson and A. Erlanson. Multi-item Vickrey–English–Dutch auctions. *Games and Economic Behavior*, 81:116–129, 2013.
[13] A. Angelo, P. Thellmann, and D. Dalkilic. Rewarding the token economy. https://bounty0x.io/whitepaper_en.pdf, 2018.
[14] O. Ascigil et al. On uncoordinated service placement in edge-clouds. In *IEEE CloudCom, 2017*.
[15] I. Baldini, , et al. Serverless computing: Current trends and open problems. In *Springer Research Advances in Cloud Computing, 2017*.
[16] F. Bonomi et al. Fog computing and its role in the internet of things. In *ACM MCC workshop, 2012*.
[17] M. Claypool and K. Claypool. Latency and player actions in online games. *ACM Communications, 2006*.
[18] R. Dua, A. R. Raja, and D. Kakadia. Virtualization vs containerization to support paas. In *IEEE IC2E, 2014*.
[19] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics*, 8(3):399–404, 1956.
[20] M. Król and I. Psaras. Nfaas: named function as a service. In *ACM ICN, 2017*.
[21] R. Landa et al. Self-tuning service provisioning for decentralized cloud applications. *IEEE TNSM*, 2016.
[22] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *ACM SIGCOMM conference on Internet measurement, 2010*.
[23] A. Madhavapeddy and D. J. Scott. Unikernels: Rise of the virtual library operating system. *Queue*, 11(11):30, 2013.
[24] D. Mishra and D. C. Parkes. Multi-Item Vickrey-Dutch auctions. *Games and Economic Behavior*, 2009.
[25] D. Mishra and D. Talman. Characterization of the walrasian equilibria of the assignment model. *Journal of Mathematical Economics*, 2010.
[26] M. Peng, S. Yan, K. Zhang, and C. Wang. Fog-computing-based radio access networks: issues and challenges. *IEEE Network, 2016*.
[27] I. Psaras et al. Keyword-based mobile application sharing. In *ACM Workshop on Mobility in the Evolving Internet Architecture, 2016*.
[28] A. Russell. Progressive web apps: Escaping tabs without losing our soul. *Infrequently Noted*, 2015.
[29] J. K. Sankaran. On a dynamic auction mechanism for a bilateral assignment problem. *Mathematical Social Sciences*, 1994.
[30] M. Satyanarayanan. The emergence of edge computing. *Computer*, 50(1):30–39, 2017.
[31] L. S. Shapley and M. Shubik. The assignment game I: The core. *International Journal of Game Theory*, 1971.
[32] J. Swearingen. http://nymag.com/selectall/2018/03/when-amazon-web-services-goes-down-so-does-a-lot-of-the-web.html, 2018.
[33] A. Tasiopoulos. *On the Deployment of Low Latency Network Applications over Third-Party In-Network Computing Resources*. PhD thesis, UCL (University College London), 2018.
[34] A. Tasiopoulos et al. Fogspot: Spot pricing for application provisioning in edge/fog computing. *IEEE Transactions on Services Computing, 2019*.
[35] A. G. Tasiopoulos et al. Edge-MAP: Auction markets for edge resource provisioning. In *IEEE WoWMoM, 2018*.
[36] A. G. Tasiopoulos et al. On-path cloudlet pricing for low latency application provisioning. In *IEEE LANMAN, 2018*.
[37] S. Uppoor, O. Trullols-Cruces, M. Fiore, and J. M. Barcelo-Ordinas. Generation and analysis of a large-scale urban vehicular mobility dataset. *IEEE Transactions on Mobile Computing, 2014*.
[38] H. R. Varian and C. Harris. The VCG auction in theory and practice. *American Economic Review, 2014*.