Edge-MAP: Auction Markets for Edge Resource Provisioning

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

***Abstract*—New and emerging applications in the entertainment (*e.g.,* Virtual/Augmented Reality), IoT and automotive domains will soon demand response times an order of magnitude smaller than can be achieved by the current "client-to-cloud" network model. Edge- and Fog-computing have been proposed as the promise to deal with such extremely latency-sensitive applications. According to Edge-/Fog-Computing, computing resources are available at the edge of the network for applications to run their virtualised instances. We assume a distributed computing environment, where *In-Network Computing Providers* (INCPs) deploy and lease edge resources, while *Application Service Providers* (AppSPs) have the opportunity to rent those resources to meet their application's latency demands. We build an *auction-based resource allocation and provisioning* mechanism which produces a map of application instances in the edge computing infrastructure (hence, acronymed *Edge-MAP*). Edge-MAP takes into account users' mobility (*i.e.,* users connecting to different cell stations over time) and the limited computing resources available in edge micro-clouds to allocate resources to bidding applications. On the micro-level, Edge-MAP relies on Vickrey-English-Dutch (VED) auctions to perform robust resource allocation, while on the macro-level it fosters competition among neighbouring INCPs. In contrast to related studies in the area, Edge-MAP can scale to any number of applications, adapt to dynamic network conditions rapidly and reallocate resources in polynomial time. Our evaluation demonstrates Edge-MAP's capability of taking into account the inherent challenges of the provisioning problem we consider.**

I. Introduction

Cloud computing has been a tremendous technological and commercial success in provisioning applications due to its essentially boundless elasticity in terms of computing resources. However, remote data centres where cloud services usually reside are associated with long network RTTs and traffic bottlenecks [24]. This can prohibit certain applications with stringent latency and bandwidth requirements—which we refer to as *"Low Latency Applications"* (LLAs)—from achieving a satisfactory Quality of Service (QoS). In particular, LLAs such as augmented reality, voice/image recognition, mobile gaming, and so on presents a challenge for the current centralised, cloud-based infrastructure [2].

As a result, there is a pressing need for alternative infrastructures to augment and complement centralised, remote datacentres, in order to enable such applications [32]. Cloudlets have been proposed as "*data centres in a box*" [33] that can be placed at the edge and middle-tier locations of the network, as shown in Fig. 1. Provisioning resources to run instances of LLAs over the cloudlet infrastructure, as opposed to cloud, brings the applications "*closer*" to the end users. This reduces the inherent cloud access latency while avoiding network bottlenecks. At the same time, the provisioning of cloudlets' resources is challenging due to their limited

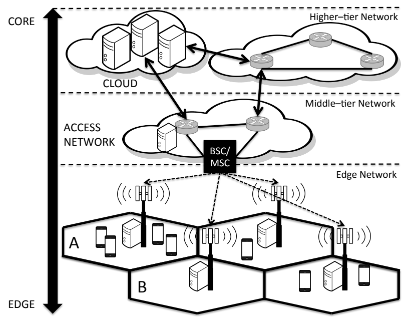


Fig. 1: Cloudlet Infrastructure Deployment

elasticity, rendering static application provisioning—*i.e.,* one where resources are statically partitioned and each partition is dedicated to a specific application—inefficient. Clearly, *how to manage the limited resources available at the distributed cluster of cloudlets* is of paramount importance in provisioning resources for LLAs.

In this work, we assume a set of geo-distributed cloudlet infrastructures, each of which is owned by an *in-network computing provider* (INCP) that is willing to lease its resources to host LLAs. On the other hand, LLAs are owned by *Application Service Providers* (AppSPs) who are willing to pay INCPs in order to run their applications in the edge infrastructure. Therefore, AppSPs provide higher QoS to their customers, while INCPs are incentivised to both maintain and expand their edge infrastructure. In this setting in order to support dynamic LLA provisioning for mobile users, we have to address the interdependent problems of:

1) Cloudlet monetary profit generation, *i.e.,* how INCPs make money, and
2) Real-time cloudlet resource management, *i.e.,* which LLA should be deployed and where.

We assume interactive applications with strict deadlines (in the order of 10-50 ms, as in the case of augmented reality [11] for example) and users that move between cells. Similarly to cloud systems, the computing resources of the cloudlets are available in the form of Virtual Machines (VMs) and are located *i)* at base-station cells and *ii)* at middle-tier network locations. Provisioning of resources at base-station cells (hexagons at the edge in Fig. 1) is particularly challenging due to users' mobility, which we explicitly consider in this work. The back-end clouds serve as the *final-call* provisioning location (Fig. 1) for user requests that fail to obtain resources at the cloudlets.

In this paper, we argue for a market-based solution that brings together INCPs who lease their resources, and AppSPs who are interested in renting edge resources to improve the QoS of their mobile users. To this end, we propose a market mechanism tailored to the mobile users demand dynamics that benefits both LLAs' QoS and INCPs' income. We introduce *Edge-MAP*, as the underlying framework that enables the *in-network, on-demand provisioning market for LLAs*. A distinct novelty of Edge-MAP, compared to related studies, is that it perceives the INCPs' resources as a *"pool of interconnected virtualised hardware"* offered via independent marketplaces located at each cell. According to this view of the network, resources physically located but under-utilised at some cell, *e.g.,* cell B in Fig. 1, can be advertised in neighbouring cell markets where demand exceeds supply, *e.g.,* cell A in Fig. 1. We assume that a Vickrey-English-Dutch (VED) auction [5] is deployed at each cellular market to provision LLAs' instances over the offered VMs in polynomial time. Edge-MAP fosters competition among INCPs (located at various cells) by providing feedback related to their VMs' value in local and remote marketplaces.

The main technical contributions of this paper are as follows:

1) We study the problem of resource provisioning over a distributed INCPs' infrastructure for LLAs.
2) We design the Edge-MAP mechanism for LLAs' provisioning in the challenging case of mobile users.
3) We apply VED auctions in problems where the demand/supply conditions evolve over time.
4) We evaluate Edge-MAP on realistic vehicle traffic patterns.

II. Edge-MAP Design and System Model

In this section we present Edge-MAP's design principles, that support *on-demand provisioning markets for mobile users*, followed by the system model description.

A. Edge-MAP's Design Principles

Application provisioning refers to the allocation of cloud resources, in the form of VMs, for serving the demand of LLAs' end users [26]. LLA provisioning over geo-distributed cloudlet resources differs from typical application provisioning in the sense that the allocation of resources has to take into account the impact of the network conditions, *i.e.,* the latency between the end users and cloudlets' points of presence [3], on applications' QoS. Moreover, in the case of mobile users, the latency to an allocated VM changes as soon as users handover their connections to another base-station. Hence, even optimal VM allocations to user requests get outdated over time; that is, users' mobility/handoffs should be followed when required by VM reassignments—a process referred to as *VM handoffs* in [33]. In static provisioning, where a static number of VMs are allocated to an application for long periods of time, VM handoffs might lead to idle VMs that could instead be used by other applications' users. Here, in order to avoid VMs' underutilisation, we argue for the need of *on-demand LLA provisioning*, where a VM for an application is instantiated *upon an end-user request* for the duration of the end-user's engagement.

On-demand LLA provisioning process consists of the stages of *i*) discovery of available resources, *ii*) resource allocation, and *iii*) resource configuration (*i.e.,* booting up of VMs) [15]. In order to obtain a responsive and efficient provisioning system for mobile users, we focus on minimising the amount of time spent for resource discovery and resource allocation. With the latest advances in virtualisation technology, the configuration overhead is greatly reduced. For instance Unikernel VMs boot in less than 100-150 ms [25]. The VM configuration overhead is expected to further reduce in the future, and therefore, *we focus on reducing the overhead of discovery and allocation of resources*.

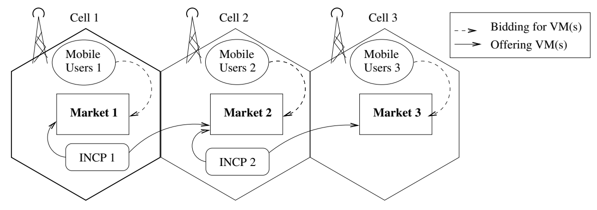


Fig. 2: VM offerings by the INCPs and user biddings for VMs in three adjacent cells.

In the competitive setting of INCPs that we envision, the discovery and allocation of resources (*i.e.,* VMs) take place in individual markets, located at the base-station of each cell, via auction mechanisms [19], [35]. In a cellular market, the offered VMs are *items*, while the mobile users, connected to the corresponding cell, are *bidders* that wish to acquire at most a single VM/item. The auction's purpose is to derive the price vector as well as the item-bidder assignments that characterise a *competitive equilibrium*. That is, the auction has to: *i*) satisfy bidders' demand for the given price vector, and *ii*) fully allocate every item with a positive price, *i.e.,* the price of each unallocated item is 0.

A distinct feature of Edge-MAP is that INCPs can offer their VMs in other cellular markets with the condition that a particular VM can only be offered uniquely at any point in time. As an example, consider Fig. 2, where three adjacent cells (numbered 1–3) are depicted along with their connected mobile users and INCPs. All cells have a dedicated market for serving their connected users while INCPs are present only in cells 1 and 2. We assume that INCP 2 offers part of their VMs to cellular markets 2 and 3 since offering VMs to adjacent cells' users leads to profit opportunities. In Edge-MAP these profit opportunities are provided by a feedback mechanism, as we explain in detail later in Section III.

Provisioning of VMs happens through periodic/discrete-time execution of auction mechanisms, where the minimum duration of each period/time-slot is restricted by the aggregated time overhead of auction execution and VM configuration, as we show in Fig. 3. In each period, the bidders adjust their demand subject to their local cell, and the associated network conditions, while the INCPs adjust their supply, in terms of offered VMs, subject to their current utilisation. That is, the objective of minimising the time overhead of resource discovery and resource allocation is equivalent to *minimising the time of accessing a market and execute an auction mechanism.*

Along these lines, the proposed Edge-MAP mechanism relies on VED auctions [5] for on-demand provisioning for the following reasons:

- VED auctions *derive the unique minimum competitive equilibrium prices* [34], known as Vickrey-Clarke-Groves (VCG) prices; that is, the bidders cannot acquire their assigned items for a lower price in any other competitive

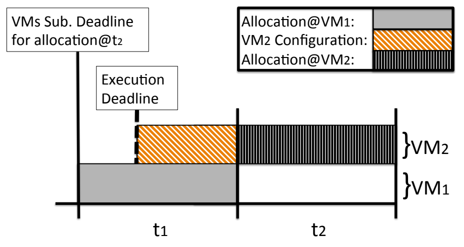


Fig. 3: Time-slot t_1 duration decomposed into *i*) the auction execution deadline, *i.e.,* the maximum expected execution time of the auction, and *ii*) the VM configuration overhead for provisioning at time-slot t_2, *e.g.,* VM_1 handoff to VM_2 at t_2.

equilibrium.
- The VED auctions result in the VCG prices *starting at any possible initial price* for each one of the items.

Because the Edge-MAP is executed repetitively, a VED auction can re-utilise previously found VCG prices and reduce the execution time of the mechanism, *e.g.,* if the supply and demand conditions of the market remains the same, the VCG prices of the new time-slot will be identical to the previous one, and the auction will terminate immediately since it starts from its equilibrium prices. Moreover, the bidders are *truthful* since they acquire an item in its minimum possible price; therefore, they do not have incentives to deploy complex strategies that would prolong the execution of the auction.

Finally, in Edge-MAP instead of considering a single centralised market, *we approach each cell as a distinct market-place, organised by a local auctioneer to serve the on-demand provisioning requests of its currently connected users*. This choice of cell-based markets comes with the advantages of:

- *Resource discovery and allocation time overhead minimisation*, since a user request is accessing a market immediately at her local cell. At the same time, the auction's execution involves a considerably smaller number of bidders, leading to a significantly lower execution time.
- *Providing profit opportunities to INCPs*, which can offer their VMs to different cellular markets at different prices. In particular, Edge-MAP provides feedback to INCPs regarding their demand in each market, fostering their competition as we explain in III-B.

1) Edge-MAP Overview: Our design, on top of the already introduced AppSPs and INCPs, involves the following players:

- *Auctioneer:* The entity that collects arriving bids for LLA provisioning and allocates INCPs' VMs to the highest set of bids. There is *exactly one auctioneer per market.*
- *AppSP Agents:* The entities that represent AppSPs in every cellular market. These entities actually bid on behalf of the mobile users for VMs offered in each market, with respect to the requested LLA requirements.

In Fig. 4, we depict the Edge-MAP's provisioning process upon the arrival of a new user in a cellular market. At first, the mobile user sends an application request to the local market which is directed to the corresponding AppSP Agent (step a). The AppSP Agent is aware of the application requirements, *i.e.,* in terms of how the latency affects its QoS. In particular, the AppSP has pre-estimated the potential gain in QoS that

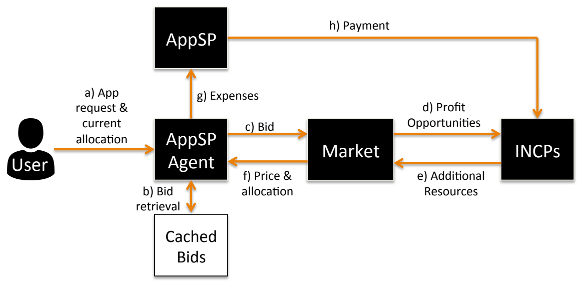


Fig. 4: Edge-MAP Overview

a VM provisioning can produce at each INCP for a specific LLA request arriving at the local cellular market. Since the VED auction is an incentive-compatible mechanism [5], the AppSP Agent acts truthfully and bids for resources by just setting the bids equal to the actual gain, in terms of QoS, of the application (steps b and c).[1] Next, the VED auction is executed followed by the INCPs' feedback phase about profit opportunities (step d) that aim to attract more VM resources to the market (step e); concluding in the VM provisioning for the user at a specific price for the next time-slot utilisation (step f). Finally, the AppSP Agent periodically informs the AppSP for her expenses at this market (step g) in order to arrange the payment of the involved INCPs (step h).

B. System Model

We consider the state of a cellular market before the execution of the auction mechanism, *e.g.,* the beginning of period t_1 in Fig. 3, that will define the LLA provisioning for the next time-slot. Let $\mathcal{S} \triangleq \{1, 2, 3, ..., S\}$ be the set of LLAs, where each LLA s is requested by $\mathcal{M}_s$ *unit demand* users, *i.e.,* interested in *at most* a single VM, that are currently connected to the cell of the market. We denote by $\mathcal{M} \triangleq \{1, 2, 3, ..., M\}$ the set of all mobile users, *i.e.,* $\mathcal{M} = \bigcup_{s\in\mathcal{S}} \mathcal{M}_s$, that *bid* for $\mathcal{H} \triangleq \{1, 2, 3, ..., H\}$ VMs, offered at the price vector $p \triangleq (p_h : \forall h \in \mathcal{H})$. The VMs are submitted to the market by a set of $\mathcal{C} \triangleq \{1, 2, 3, ..., C\}$ INCPs, where each $c \in \mathcal{C}$ contributes $\mathcal{H}_c$ VMs, such that $\mathcal{H}_c \subseteq \mathcal{H}$ and $\mathcal{H}_c \cap \mathcal{H}_{c'} = \emptyset$ for all $c, c' \in \mathcal{C}$ when $c \neq c'$.

We assume that each AppSP Agent is aware of the latency between the current cell and the location of nearby INCPs. Based on the latency information, the AppSP Agent of a LLA $s \in \mathcal{S}$, derives the expected QoS produced by provisioning a VM at INCP $c \in \mathcal{C}$ for serving an s LLA request for the next time-slot, $u_{s,c}$. Furthermore, assuming an always available default provisioning (*i.e.,* back-end cloud) infrastructure for LLA s, the AppSP Agent can similarly estimate the default expected QoS, $u_{s,\emptyset}$. Then, the valuation (*i.e.,* QoS gain) of a user $m \in \mathcal{M}_s$ with respect to a VM $h \in \mathcal{H}_c$ is:

$$v_{m,h} = u_{s,c} - u_{s,\emptyset}. \tag{1}$$

Note that $v_{m,\emptyset} = 0$. The notation used throughout the paper is given in Table I.

[1]These bids can be customised to the needs of each mobile user by incorporating the VM migration cost, given the user current allocation, a case that we do not consider in detail here.

TABLE I: Model Notation

$\mathcal{S}$	Set of LLAs.
$\mathcal{M}, \mathcal{M}_s$	Set of users connected to the cell, Users requesting LLA s.
$\mathcal{C}$	Set of INCPs.
$\mathcal{H}, \mathcal{H}_c$	Set of offered VMs, Set of offered VMs by INCP c.
$\tilde{\mathcal{H}}$	Universally allocated VMs.
p, p_h	Vector price of offered VMs, Price of VM h.
$u_{s,c}$	QoS produced by serving a request of LLA s at INCP c.
$u_{s,\emptyset}$	QoS produced by serving LLA s at its default location.
$v_{m,h}$	Valuation of user m for VM h.
$D_m(p)$	Demand correspondence of user m function to price vector p.
$\tilde{S}, S^*, E^*$	Set of VMs in positive excess demand, excess supply, and excess demand.
p^{VCG}	VCG price vector of offered VMs.

III. Edge-MAP Mechanism

In this section, we introduce Edge-MAP's micro- and macro- level operating components of:

1) **Edge-MAP Cellular:** That is deployed in each cellular market to perform the on-demand provisioning of LLAs.
2) **Edge-MAP Orchestrator:** That provides feedback to each INCP regarding profit opportunities of its over-demanded VMs in each cellular market.

A. Edge-MAP: Cellular Operation

We describe Edge-MAP's micro-level operating component by focusing on a single cellular market. At first, we define the set of over-demanded/supplied VMs that characterise any *Multi-item auction with unit demand bidders*. After that, we provide details about how VED auctions derive the VCG equilibrium by increasing (decreasing) the price of the VMs that are considered over-demanded (over-supplied) in the context of Edge-MAP mechanism.

1) VMs in Excess Supply and Excess Demand: We consider the extended market of $\mathcal{H}^* = \mathcal{H} \cup \{\emptyset\}$ VMs, where the *null item* $\{\emptyset\}$ corresponds to the default VM provisioning of each request at the corresponding back-end cloud.[2] Let p_h be the price of VM $h \in \mathcal{H}^*$ and $v_{m,h}$ be the valuation of bidder/user m for VM h (Eq. 1). The *demand correspondence* of user m is defined as:

$$D_m(p) \triangleq \{h \in \mathcal{H}^* : v_{m,h} - p_h \geq v_{m,h'} - p_{h'}, \forall h' \in \mathcal{H}^*\} \quad (2)$$

in other words, the $D_m(p)$ set includes the VMs that maximise the user's valuation after the price reduction, known as net-valuation, *i.e.,* $v_{m,h} - p_h$. Note that at the default provisioning location the VMs' price equals to zero, *i.e.,* $p_\emptyset = 0$. In Fig. 5, a market of four VMs and four users is depicted forming the following demand correspondence sets: $D_{m_1}(p) = \{h_1\}$, $D_{m_2}(p) = \{h_2\}$, $D_{m_3}(p) = \{h_2, h_3\}$, and $D_{m_4}(p) = \{h_3\}$.[3] We use the example of Fig. 5 as a point of reference in the upcoming definitions.

Given the users' demand correspondence to a price vector, the *universally allocated items*, $\tilde{\mathcal{H}}$, are defined as the set of VMs which either have a price equal to 0 or satisfy at their current price at least 2 bidders, *i.e.,* $p_h = 0$ or $\exists m, m' \in \mathcal{M}$: $h \in D_m(p) \cap D_{m'}(p)$ where $m \neq m'$ for each $h \in \tilde{\mathcal{H}}$. That is, in the example of Fig. 5 the set of universally allocated items is $\tilde{\mathcal{H}} = \{h_2, h_3\}$. Then for a set of universally allocated items, authors in [5] define the set of *positive excess demand*, $\tilde{S}$, as the universally allocated items/VMs with positive price, *i.e.,* $\tilde{S} \triangleq \{h \in \tilde{\mathcal{H}} : p_h > 0\}$. In the example of Fig. 5 for $\tilde{\mathcal{H}} = \{h_2, h_3\}$, the positive excess demand set is $\tilde{S} = \{h_2\}$ since $p_{h_2} > 0$ while $p_{h_3} = 0$. Furthermore, they prove that set $\tilde{S}$ can be identified in polynomial time by the FindUnivAllocItems procedure presented in [31].

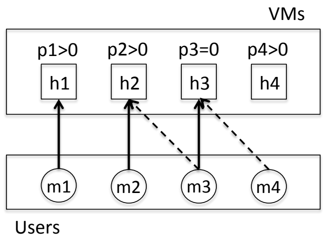


Fig. 5: VMs assignment example.

On the other hand, the set of **excess supply**, S^*, is defined as the set of *not* universally allocated items/VMs with positive price, *i.e.,* $S^* = \{h \in \mathcal{H} : p_h > 0\} \setminus \tilde{S}$; meaning, that in the example of Fig. 5 the excess supply set is $S^* = \{h_1, h_4\}$ since both VMs have a positive price while not belonging to the set of positive excess demand. Finally, the concept of the **excess demand** set is introduced. Intuitively, a set of VMs E is in excess demand at a given price vector, if *i*) the number of VMs in each proper subset T of E, $T \subset E$, is strictly smaller than the number of users that demand a VM in T, and *ii*) the users that demand at least a VM in E do not request VMs outside E. Furthermore, in [4] the authors prove that there exists a unique set in *excess demand with maximal cardinality* E^* that can be identified in polynomial time by using the "Ford and Fulkerson" algorithm, presented in [17]. In the example of Fig. 5, $E^* = \{h_2, h_3\}$.

2) VCG Equilibrium Derivation: In this section, we explain how Edge-MAP mechanism applies VED auctions on micro-level to reach the VCG equilibrium of the cellular market. VED auctions can derive the VCG equilibrium prices, p^{VCG}, as well as the corresponding users to VMs assignment, *i.e.,* $x^{\text{VCG}} : \mathcal{M} \to \mathcal{H}$, starting from any initial price vector p.[4] At a given time-slot, *e.g.,* time-slot t_2 in Fig. 3, Edge-MAP exploits VED auctions by initialising the price of each VM in the market according to the p^{VCG} solution found in the previous time-slot, *e.g.,* t_1. At the core of VED auctions is the elimination of the VMs in excess demand, E^*, and excess supply, S^*, since they are associated to the price vector p^{VCG} by the following theorem, proved in [28]:

Theorem 1. A price vector, p, equals the VCG price vector, p^{VCG}, if and only if the sets of excess demand and excess supply are empty, *i.e.,* $E^* = \{\emptyset\}$ and $S^* = \{\emptyset\}$.

Therefore, VED auctions eliminate sets E^* and S^* iteratively by updating the price of each VM h at the k-th iteration, p_h^k, according to:

$$p_h^k = \begin{cases} p_h^{k-1} + \Delta p, & if\ h \in E^{*,k-1}, \\ p_h^{k-1} - \Delta p, & if\ h \in S^{*,k-1}, \\ p_h^{k-1} & otherwise. \end{cases}$$

where if $h \in E^{*,k-1}$ (*i.e.,* eliminating the excess demand set upon iteration $k-1$), the price is increased by Δp; on the

[2]The null item is allocated to requests as many times as necessary; that is, there is no request without an assigned VM.

[3]We do not depict the null element.

[4]Other Multi-item auctions like the ascending (descending) Vickrey-English (Vickrey-Dutch) auctions [27], [31] require to start their execution from the lowest (highest) possible price in the market for each item.

other hand, if $h \in S^{*,k-1}$ (*i.e.,* eliminating the excess supply set upon iteration $k-1$), the price is decreased by Δp. The policy of increasing the prices of each VM in E^* is known as E^**-increase* while the policy of decreasing the prices in S^* is known as S^**-decrease*.

Lemma 1: Consecutive applications of E^**-increase* (S^**-decrease*) policy eliminate the excess demand E^* (excess supply S^*) set in polynomial time.

Proof: There is a maximum price that any VM in the market could be allocated $\bar{p} = \max_{\forall m \in \mathcal{M}, h \in \mathcal{H}} v_{m,h}$. Thus, the E^**-increase* policy takes at most $\lceil \bar{p}/\Delta p \rceil$ steps to exclude every VM in E^* from the users' demand correspondence, by increasing their price as their net valuation $v_{m,h} - p_h$ declines; resulting, in the elimination of E^*. Similarly, for the S^**-decrease* policy it takes at most $\lceil \bar{p}/\Delta p \rceil$ steps for every VM in S^* to have a price equal to 0, eliminating S^*. During the E^* (S^*) elimination process the "Ford and Fulkerson" algorithm (FindUnivAllocItems procedure) is called to identify the E^* (S^*) at most $\lceil \bar{p}/\Delta p \rceil$ times; therefore, since the "Ford and Fulkerson" algorithm (FindUnivAllocItems procedure) runs in polynomial time the elimination process is also polynomial. ■

However, applying both E^**-increase* and S^**-decrease* policies at the same iteration might trap the process in 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 [27]. Therefore, the E^**-increase* (S^**-decrease*) policy has to be applied in isolation in each iteration until completely eliminating E^* (S^*) before changing to policy S^**-decrease* (E^**-increase*) targeting the set S^* (E^*). The convergence to a price vector where both sets of excess supply and excess demand are empty, $E^* = S^* = \{\emptyset\}$, is guaranteed by the following monotonicity lemma proved in [5]:

Lemma 2: For any price vector $p > 0$, i) If $S^{*,k} = \{\emptyset\}$ and an E^**-increase* price adjustment policy is applied at iteration k, then $S^{*,k+1} = \{\emptyset\}$; similarly, ii) if $E^{*,k} = \{\emptyset\}$ and an S^**-decrease* price adjustment policy is applied at iteration k, then $E^{*,k+1} = \{\emptyset\}$.

A typical iteration of VED auction is presented in procedure "VED-Iteration" of Algorithm 1. Essentially the procedure applies a E^**-increase* policy (line 24) as long as E^* is not empty; otherwise, an S^**-decrease* policy is applied (line 21) until set S^* is empty too and the VCG equilibrium has been found (lines 18-19).

B. Edge-MAP: Orchestrator Operation

Edge-MAP Orchestrator component, is responsible for providing sufficient information to the INCPs participating in a market about profit opportunities. In that way, Edge-MAP aims to act beneficially for both AppSPs and INCPs by promoting the competition between INCPs, who can develop their own VM supply strategies over different cells/markets in their proximity. First of all, VMs offered by the same INCP are identical from the user perspective since her valuation is specific to the cloudlet location; therefore, she has no preference between two VMs that coexist at the same cloudlet. The following lemma associates the price of identical VMs with the set of VMs in excess demand.

Lemma 3: Identical VMs, in terms of users' valuation, can only co-exist in the set of excess demand, E^*, if their prices are equal.

```
Data: $p$, $\mathcal{M}$, $\mathcal{H}$,$\Delta p$,$\Delta|h|$
. Result: $p^{\mathrm{VCG}}$,$x^{\mathrm{VCG}}$,
2 Initialisation: $k = 1, p^1 = p$.
3 while True do
4     Collect $D_m(p^k)$ $\forall m \in \mathcal{M}$
5     Estimate $E^{*,k}$,$x^k$.
6     if ($E^{*,k} \neq \emptyset$) then
7         $\mathcal{H}'$:=$E^*$-dimensioning($E^{*,k}$, $p^k$,$\mathcal{H}$,$\Delta|h|$).
8         $\mathcal{H} = \mathcal{H} \cup \mathcal{H}'$
9     $p^{k+1}, x^{k+1}, flag$:=VED-Iteration($p^k$, $\mathcal{M}$,$\mathcal{H}$,$\Delta p$)
10    if (flag) then
11        return: $p^{k+1}$,$x^{k+1}$
12    $k$+:=1
13 end
14 VED-Iteration($p^k$, $\mathcal{M}$,$\mathcal{H}$,$\Delta p$)
15    Initialisation: Collect $D_m(p^{b,k})$, $\forall m \in \mathcal{M}$,
          Estimate $E^{*,k}$,$x^k$,$p^{k+1} = p^k$.
16    if ($E^{*,k} == \emptyset$) then
17        Estimate $S^{*,k}$.
18        if ($S^{*,k} == \emptyset$) then
19            return: $p^{k+1}$,$x^k$, True.
20        else
21            $p_h^{k+1} = p_h^k - \Delta p$, $\forall h \in S^{*,k}$
22        end
23    else
24        $p_h^{k+1} = p_h^k + \Delta p$, $\forall h \in E^{*,k}$
25    end
26    return: $p^{k+1}$,$x^k$, False.
```

Algorithm 1: Edge-MAP on demand provisioning mechanism.

Proof: Given the way that the demand correspondence, $D_m(p)$, is defined in Eq. (2), the users show a preference among identical VMs with the same *valuation*, v, for the one with the smallest price, p^*, since it maximises their *net-valuation*, *i.e.,* $v - p^*$. Therefore, the only VM that could belong to E^* set is the one with the smallest price. ■

Next, consider an INCP managing a cloudlet that participates at a cellular market at time-slot t with 10 VMs. If all of the VMs are in excess demand, then according to Lemma 3, the VMs have the same price p_h. Then Edge-MAP gives to the INCP the options of i) waiting for the E^**-increase* policy to increase the p_h price by Δp and go to the next VED auction iteration, or ii) increasing the number of the VMs participating in the market of the cell by at most $\Delta|h|$ additional VMs. We refer to the second option as E^**-dimensioning* policy (Algorithm 1 line 7). In other words, via E^**-dimensioning* identical VMs to the ones in excess demand are *supplied* into the cellular market in order to eliminate the excess demand set as we show next.

Lemma 4: The combined application of E^**-dimensioning* and E^**-increasing* policies in a single iteration of Algortihm 1, eliminates the excess demand set, E^*, in polynomial time.

Proof: Assume that $|\tilde{\mathcal{M}}|$ number of users request VMs from E^*, *i.e.,* $|\tilde{\mathcal{M}}| > E^*$. Then if there are available VMs to be offered by the INCPs, the E^**-dimensioning* policy requires $|\tilde{\mathcal{M}}| - |E^*|$ steps, when $\Delta|h| = 1$, or only 1 step, when $\Delta|h|=|\tilde{\mathcal{M}}|-|E^*|$, to eliminate the excess demand E^*, since the number of VMs in E^* will no longer be less than the number of users. However, in the worst case the "Ford and Fulkerson" algorithm is called to identify set E^* in each iteration of Algortihm 1 twice, once for the E^*-*dimensioning* and once for the E^*-*increasing* policy. But again the identification of set E^* is bounded by $2\lceil\bar{p}/\Delta p\rceil$; that is, E^* elimination takes place in polynomial time. ∎

For example, in Fig. 5 where the set of excess demand is $E^* = \{h_2, h_3\}$, if the E^*-*dimensioning* policy introduce a VM h' that is identical to h_2, the excess demand set is immediately eliminated as we see in Fig. 6. Essentially, an INCP receives information about her VMs price and number in excess demand in a specific market. Then by applying her own profit maximisation strategy, the INCP estimates a minimum price that she would accept for contributing additional VMs, denoted by $p_{\min}$. If $p_{\min}$ is higher than the current market price, p_h, the INCP will wait for the E^*-*increase* policy to be applied and increase the p_h price. On the other hand, if $p_{\min} < p_h$ she will supply this market with additional VMs. Nevertheless, elaborating on the INCPs' strategies is beyond the scope of this work. Note that Edge-MAP allows the application of the E^*-*dimensioning* policy for an INCP under the conditions that *a*) all of the INCPs' VMs are in excess demand, and *b*) the VMs to be included to the cell market are not currently involved in any other market.

Theorem 2. Edge-MAP mechanism, as described in Algorithm 1, converges to the VCG equilibrium in polynomial time.

Proof: From Lemma 4 we have that the initially applied E^*-*dimensioning* and E^*-*increase* policies in Algorithm 1 eliminate E^* in polynomial time. After the E^* elimination, the S^*-*decrease* policy eliminates the excess supply S^* in polynomial time (Lemma 1) without affecting the already eliminated E^* set (Lemma 2). Thus from Theorem 2, Algorithm 1 derives the VCG equilibrium after 2 sequential polynomial time processes; rendering Algorithm's 1 execution time polynomial. ∎

IV. Performance Evaluation

In this section, we demonstrate the performance of Edge-MAP. We begin by describing the setup of our evaluation before presenting our results.

A. Evaluation Setting

Mobility traces and cellular setting: The evaluation is based on a mobility dataset[5] developed in the context of TAPASCologne project by using a state of the art mobility generator tool [36]. The dataset consists of 700,000 car journeys covering a region of 400 square kilometres over 24 hours. We construct a cellular infrastructure by dividing the region into 864 hexagon cells of 1Km radius for each cell. We consider each vehicle as a mobile user, associated with a mobile device, and we focus on a single off-peak hour of the dataset, *i.e.,* 11am, which presents a good ratio of average vehicle speed and number of vehicles moving at each second, *i.e.,* ~30 Km/h and ~5200 vehicles, respectively. During this period the population remains relatively constant, with approximately 13 vehicles leaving/arriving every second. The number of vehicles per cell for a snapshot of the data, at 11am, is shown in Fig. 7.

[5]http://kolntrace.project.citi-lab.fr

TABLE II: Default Evaluation Setting

Cells Distance Separation	1 Km
Number of VMs per Cloudlet	20
Number of LLA categories	10
Δp, $\Delta\lvert h\rvert$	1
$p_{\min}$	0
Time-slot duration	60s
Per hop in between cell latency	10 ms
INCP market participation in cellular hops	1

LLA categories: The QoS of the LLAs is expressed as a decreasing function of the end user perceived round-trip time (RTT), in terms of latency x, to each cloudlet [13], [38]. Similarly to [20] for abstract resource allocation gains, we consider that each LLA is characterised by a decreasing QoS function of the general form:

$$u(x) = \left(\frac{u_{\min}}{u_{\max}} + \left(1 - \frac{u_{\min}}{u_{\max}}\right)\left(1 - \frac{x - l_{\min}}{l_{\max}}\right)^{\frac{1}{\alpha}}\right) \times u_{\max} \quad (3)$$

The constants $u_{\max}$ ($u_{\min}$) represents the maximum (minimum) QoS that the application user can achieve at the minimum (maximum) latency $l_{\min}$ ($l_{\max}$), *i.e.,* $u(l_{\min}) = u_{\max}$ and $u(l_{\max}) = u_{\min}$. We set $u_{\max} = 100$, $l_{\min} = 4$ ms[6], and $l_{\max} = 500$ ms assuming that all applications have identical latencies to their default cloud locations[7]. Moreover, function $u(\cdot)$ is convex for $0 < \alpha \leq 1$; that is, we set $\alpha = 0.2$ since LLAs' QoS is expected to be more sensitive to latency changes closer to $l_{\min}$. We depict the QoS function for α values 0.2 and 1.0 in Fig. 8.

Based on Eq. 3, we create ten *LLA categories*, each associated with a QoS function that models a certain sensitivity of the LLA to network latencies. We use a different $u_{\min}$ value assigned $\{0, 10, 20, \dots, 90\}$ for each category of LLA. In this way, different application categories have different gains from being provisioned at the edge, varying from 10, for $u_{\min} = 90$ and less latency sensitive LLAs, to 100, for $u_{\min} = 0$ and latency critical LLAs. We consider the number of ten LLA categories sufficient for the purpose of our evaluation since it is comparable to the currently considered types in the context of IoT [10], [23] and Tactile Internet [16].

Setting parameterisation: In the mobile environment we consider, we focus *only* on cloudlets located at the network edge, *i.e.,* base-station cells. In the default setting, cloudlets are allowed to advertise and offer their VMs to cellular markets that are up to 1 cell away, covering the cells denoted by r_0 and r_1 in Fig. 9 when the cloudlet is located at r_0. We set the inter-cell latency to 10 ms and assume a tree-like backhaul topology [29], where the latency between cells increases linearly with the hop distance, *e.g.,* if a cloudlet is located 2 cells away from the cell a user is connected to, then the involved latency is 20ms. Given the network topology latencies, we round up the non-integer QoS values produced by Eq. 3, we parametrise Edge-MAP mechanism by setting $\Delta p = 1$, $\Delta|h| = 1$, $p_{\min} = 0$, and we set the time-slot duration equal to 60s.

[6]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

[7]500 ms is the maximum latency observed for Amazon Web Services according to CloudPing, available at http://www.cloudping.info.

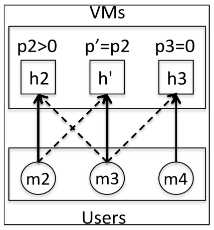


Fig. 6: E^**-dimensioning* example.

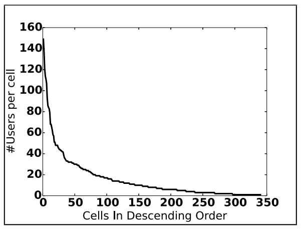


Fig. 7: Users per cell in descending order, at 11am.

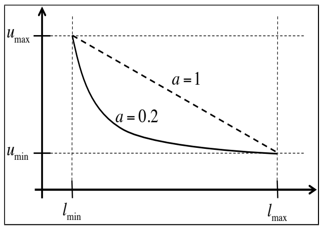


Fig. 8: QoS function we consider for a LLA category.

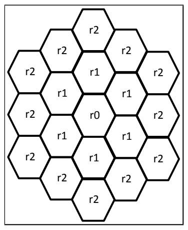


Fig. 9: Allowed Market Participation of a cloudlet at r_0.

Considering the statistics provided by the Smart Insights' [1] survey regarding *i*) the user application engagement duration over a day, and *ii*) the smartphones market penetration percentage, we consider that the 10% of the mobile users are engaged to a LLA at any second, *i.e.,* 520 users on average. We set the capacity of each cloudlet to 20VMs, which is on average sufficient for serving even the most crowded cells given the 10% users' participation we consider. Note that we limit each cell to host *at most a single cloudlet* in all the experiments. The default values of the experiment parameters are provided in Table II.

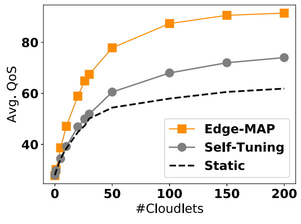
LLA requests generation: We consider two approaches of generating service requests, namely:

- *Probabilistic requests*: Each user selects one of the 10 LLA categories according to a Zipf distribution that favours the most QoS sensitive application categories, *i.e.,* the most popular applications are the ones with the highest $u(l_{\min})$. Similarly to edge caching systems [9] we set the Zipf's distribution exponent equal to 0.8.
- *Realistic requests*: Each user selects one of the 10 LLA categories based on the sequence of request arrivals in Google's cluster dataset[8]. In detail, we associate each LLA category to a "ParentID" field, that identifies the service, of the 10 most popular services in the dataset accounting for more than 200K requests. Then by selecting random time intervals in the period of the seven hours that the dataset covers, users request LLA categories based on the sequence of "ParentID" fields that arrive into Google's cluster.

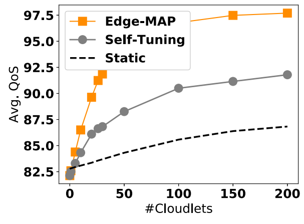
In both cases, we assume that the users remain engaged to their requested LLA throughout their journey.

Cloudlet deployment: We evaluate the impact of the Edge-MAP in relation to the availability of cloudlet resources, which are incrementally deployed over the cells starting from the most crowded ones. In this way, we capture Edge-MAP's behaviour over the spectrum of different cloudlets' infrastructure conditions; starting from under-deployed, where a cloudlet exists only at the most crowded cell, to over-deployed, where there is an installed cloudlet at each cell. The turning point between the over- and under- deployed infrastructure is taking place upon the deployment of the 26th cloudlet, where the available VMs, *i.e.,* 26×20=520 VMs, equals the average number of participating users, *i.e.,* 10% of the 5200 users in the dataset. Note that all cells support a marketplace no matter if a cloudlet is locally deployed.

[8]Available at https://research.googleblog.com/2010/01/google-cluster-data.html.



(a) Probabilistic Requests



(b) Realistic Requests

Fig. 10: Edge-MAP vs. Static, and Self-Tuning Provisioning

B. Simulation Results

The results presented next have been averaged over 100 executions.

Impact of on-demand provisioning: We compare Edge-MAP against *i*) Static and *ii*) Self-Tuning [22] provisioning in terms of average QoS. In static provisioning, we assume that each AppSP is aware of her aggregate service demand and allocates a portion of VMs at each cloudlet that corresponds to this demand. For example, given the cloudlet capacity of 20 VMs and a LLA $s \in \mathcal{S}$ that accounts for half of the generated requests, static provisioning will allocate statically 10 instances of s at each cloudlet for the entire duration of the simulation.

On the other hand, in self-tuning approach the provisioning of LLAs takes place periodically, repeating at regular time-slots, at each cell. The provisioning of LLAs in self-tuning approach at a time-slot t is based on *i*) the demand of each LLA observed during the time-slot $t-1$ and *ii*) the QoS gain of LLAs from being provisioned at the edge. For instance, if LLA s was requested on average by 7 users throughout time-slot t at a specific cell, the AppSP will bid for up to 7 VMs at the respective cellular market at time-slot $t+1$. If LLA s is the service with the highest QoS gain it will allocate seven VMs at the price of the next seven bids, *i.e.,* generalised second price auction. Note that the original self-tuning approach proposed in [22] is a generalised second price combinatorial auction requiring an offline execution. In order to create an online distributed variation of the self-tuning approach that it is comparable to Edge-MAP, we limit cellular markets in offering VMs that are located only at the current cell. That is, we eliminate the combinatorial difficulty of the problem by offering *identical* VMs in terms of QoS gain, since the gain of each AppSP increases linearly with the number of allocated VMs until reaching its expected demand.

Edge-MAP outperforms the other approaches in terms of average QoS in both probabilistic and realistic request generation settings, while self-tuning approach is superior only to the static provisioning approach, as we see in Fig. 10.

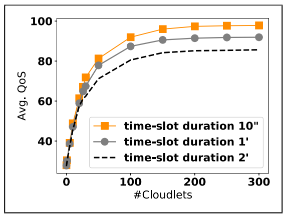


Fig. 11: Time-slot Duration QoS Impact

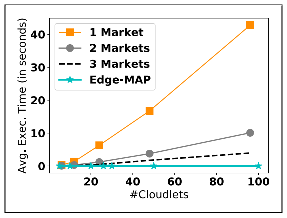
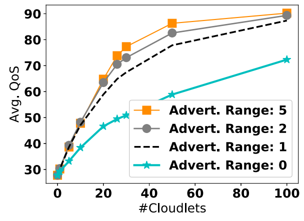
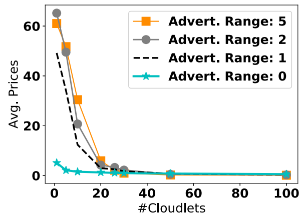


Fig. 12: Centralised Markets: Execution Time Impact.



(a) Avg. QoS.



(b) Avg. Price.

Fig. 13: Local vs. Pool of Virtual Resources

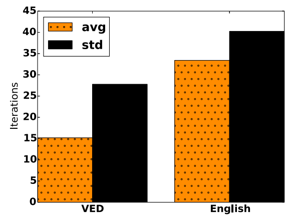
Clearly, the average QoS is lower for the Zipf-based probabilistic request generation due to the correlation between each service's popularity and QoS gain, *i.e.,* the most demanding service is the most popular, and lack of resources results in the faster deterioration of the average QoS. In the remaining experiments due to space limitation we present the results of the probabilistic request generation which is a more challenging case.

Impact of time-slot duration: In Fig. 11 we depict the average QoS under different time-slot durations, namely 10, 60, and 120 seconds, capturing an increasing configuration time overhead related to the VM management and potential applications state migration. As expected, a longer time-slot duration leads to QoS deterioration due to the decrease in Edge-MAP provisioning responsiveness to LLA demand changes, caused by mobile users' *i*) handovers and *ii*) arrivals/departures from the system.

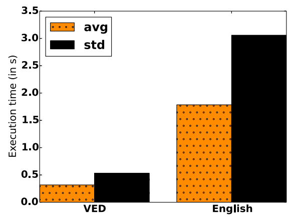
Benefit of per-cell markets: Fig. 12 demonstrates the increase in the execution time of the mechanism when the provisioning of resources takes place via a fixed number of markets instead of deploying one marketplace at each cell. In particular, we consider a centralised scenario, *i.e.,* a single market in the system, a scenario with two markets, *i.e.,* each market is responsible for roughly half of the cells in the system, and three markets, *i.e.,* each market is in charge of one third $(1/3)$ of the cells in the system. The reason behind Edge-MAP's negligible execution time compared to the fixed number of markets is that the cellular based markets involve a considerably smaller number of bidders, since they include only the users that are connected to the current base station, as well as number of VMs. Note that Edge-MAP scales gracefully as the number of cloudlets in the system increases while in the case of fixed markets the time increases linearly, due to the polynomial execution time nature of VED auctions.[9]

Benefit of interconnected virtualised resources: Fig. 13a depicts the QoS gain from allowing cloudlets to offer their VMs to distant markets, defined by the cellular advertisement range with respect to the location of the cloudlet offering its

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



(a) Iterations Comparison.



(b) Execution Time Comparison.

Fig. 14: VED vs. English Auctions

resources. Consider the cell r_0 at the center in Fig. 9, where the hexagons tagged r_1 and r_2 are within advertisement range one and two from r_0, respectively. For an advertisement range of zero, the INCP at cell r_0 can advertise its resources only at r_0's *local* market for only to be utilised by the users connected to r_0. With a range of one, on the other hand, the INCP at r_0 can now advertise its resources at the markets of r_1 cells in addition to its local one at r_0, which makes INCP r_0's resources accessible by the users connected to r_0 and r_1 cells.

Undoubtedly, there is a higher QoS gain when INCPs act as a pool of interconnected virtualised resources that can be offered over different cells than offering their resources only to the local cellular market. The reason is that idle VMs have the opportunity of being utilised by users connected to a different cellular markets, where due to the *limited elasticity* of the local cloudlet there are not available VMs for serving their requests. Furthermore, from Lemma 3 we know that VMs offered by a single cloudlet to a single market can be allocated only if they have identical prices. In other words, the price that a INCP can get from her resources is *market specific*. Therefore, offering VMs to distant markets is beneficial for the income of the INCPs since they have the opportunity to diversify their prices. This is clear in Fig. 13b, where over the under-deployed zone, *i.e.,* number of cloudlets 1 to 26, the average price of VMs is substantially higher than the case when VMs are only offered locally. On the other hand, at the over-deployed zone, *i.e.,* number of cloudlets 26 onwards, the average price is approaching the value of zero, due to the abundance of resources and the competition conditions that are created.

Benefit of VED auctions: Lastly, in order to demonstrate Edge-MAP's scalability in increasing workloads, we consider the extreme case where all mobile users request a LLA in a setting where 26 cloudlets are deployed, each one capable of supporting 200 VMs. Figures 14a and 14b present the comparison of Edge-MAP against an Edge-MAP's variation that relies on Vickrey-English (VE) ascending auctions instead of VED. Clearly, VED auctions dominate over the VE ones both in terms of iterations and execution time, since VED requires less than a second to derive the new VCG equilibrium as opposed to VE whose execution time might exceed the 4 seconds threshold. Therefore, VED auctions are ideal for repetitive allocation settings where they can take advantage of the previously found equilibrium for decreasing their execution time.

V. Related Work

Cloudlets have been proposed in [33] as a surrogate infrastructure where mobile devices could offload intensive tasks to complement their computing capabilities and battery limitations [7], [14]. Since then a considerable amount of research has focused on task offloading technologies [12], [33] targeting either the augmentation of mobile devices computing capabilities [30], [37] and/or battery duration [8]. In our previous work [6], we address the problem of application-specific task offloading (*i.e.,* a task requires the corresponding virtual instance of the application stored at a cloudlet before getting offloaded), but do not take into account user-mobility and focus instead on efficient allocation of services to computation nodes across the *edge-to-core* path. Lastly, other approaches considered the maximisation of admitted task volume [18], while task offloading from a competitive perspective is presented in [21]. Specifically, authors in [21] present a double auction scenario between end users and cloudlets but their centralised market raises questions about the scalability of such a system. Furthermore, the execution frequency of the auction mechanism is not investigated since the users' demand is considered static. Closer to our work, Landa *et al*. proposed a self-tuning service provisioning auction mechanism over the in-network computational resources that are organised in execution zones [22]. However, their solution does not achieve fine grained utilisation of resources since it is based on an offline execution that relies on each LLA's demand expectation. To the best of our knowledge, *we are the first to define a polynomial time market mechanism for LLA provisioning over an edge-computing infrastructure taking into account end user mobility and fostering competition among INCPs.*

VI. Conclusions

We studied the emerging market of LLAs' provisioning over edge-/fog-computing. Along these lines we proposed Edge-MAP, a polynomial time mechanism tailored to the on-demand provisioning of LLAs for mobile users. At the micro-level, Edge-MAP operates on cellular-based markets using VED auctions to perform robust resource allocation. At the macro-level, Edge-MAP fosters competition among INCPs by providing feedback with respect to profit opportunities on different markets. Our evaluation verified Edge-MAP's design capability to take into account the inherent challenges of LLAs and allocate resources according to demand, adapting to network conditions and avoiding underutilisation of edge-computing infrastructure.

Acknowledgment

This work has been supported by the EC H2020 UMOBILE project (GA no. 645124), the EC H2020 ICN2020 project (GA no. 723014), the EPSRC INSP fellowship (EP/M003787/1).

References

[1] Smart insights survey: http://www.smartinsights.com/mobile-marketing/mobile-marketing-analytics/mobile-marketing-statistics.

[2] S. Abolfazli et al. Cloud-based augmentation for mobile devices: motivation, taxonomies, and open challenges. *IEEE Communications Surveys & Tutorials, 2014.*

[3] S. Agarwal et al. Volley: Automated data placement for geo-distributed cloud services. In *NSDI, 2010.*

[4] T. Andersson, C. Andersson, and A. Talman. Sets in excess demand in simple ascending auctions with unit-demand bidders. *Springer, Annals of Operations Research, 2013.*

[5] T. Andersson et al. Multi-item vickrey–english–dutch auctions. *Games and Economic Behavior*, 2013.

[6] O. Ascigil et al. On uncoordinated service placement in edge-clouds. In *IEEE CloudCom, 2017 IEEE.*

[7] R. K. Balan et al. Simplifying cyber foraging for mobile devices. In *Mobile systems, applications and services*. ACM, 2007.

[8] M. V. Barbera et al. To offload or not to offload? the bandwidth and energy costs of mobile cloud computing. In *INFOCOM*. IEEE, 2013.

[9] E. Bastug, M. Bennis, and M. Debbah. Living on the edge: The role of proactive caching in 5g wireless networks. *IEEE, Communications Magazine, 2014.*

[10] F. Bonomi et al. Fog computing and its role in the internet of things. In *ACM, MCC workshop, 2012.*

[11] D. M. Chen et al. Streaming mobile augmented reality on mobile phones. In *Mixed and Augmented Reality, 2009. ISMAR 2009. 8th IEEE International Symposium on.*

[12] B.-G. Chun et al. Clonecloud: elastic execution between mobile device and cloud. In *Computer systems*. ACM, 2011.

[13] M. Claypool and K. Claypool. Latency and player actions in online games. *Communications of the ACM, 2006.*

[14] E. Cuervo et al. MAUI: making smartphones last longer with code offload. In *Mobile systems, applications, and services*. ACM, 2010.

[15] P. T. Endo et al. Resource allocation for distributed cloud: concepts and research challenges. *IEEE Network*, 25(4), 2011.

[16] G. P. Fettweis. The tactile internet: Applications and challenges. *IEEE, Vehicular Technology Magazine, 2014.*

[17] L. R. Ford and D. R. Fulkerson. Maximal flow through a network. *Canadian journal of Mathematics, 1956.*

[18] D. T. Hoang et al. Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *WCNC*. IEEE, 2012.

[19] W. Iqbal et al. Adaptive resource provisioning for read intensive multi-tier applications in the cloud. *Elsevier Future Generation Computer Systems, 2011.*

[20] H. Izakian, A. Abraham, and B. T. Ladani. An auction method for resource allocation in computational grids. *Elsevier, Future Generation Computer Systems, 2010.*

[21] A.-L. Jin et al. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. 2015.

[22] R. Landa et al. Self-tuning service provisioning for decentralized cloud applications. *IEEE TNSM*, 2016.

[23] I. Lee and K. Lee. The Internet of things: Applications, investments, and challenges for enterprises. *Elsevier, Business Horizons, 2015.*

[24] A. Li, X. Yang, S. Kandula, and M. Zhang. Cloudcmp: comparing public cloud providers. In *ACM IMC, 2010.*

[25] A. Madhavapeddy, T. Leonard, M. Skjegstad, T. Gazagnaire, D. Sheets, et al. Jitsu: Just-in-time summoning of unikernels. In *NSDI, 2015.*

[26] P. Mell, T. Grance, et al. The nist definition of cloud computing.

[27] D. Mishra et al. Multi-item vickrey–dutch auctions. *Games and Economic Behavior*, 2009.

[28] D. Mishra et al. Characterization of the Walrasian equilibria of the assignment model. *Journal of Mathematical Economics*, 2010.

[29] M. Peng et al. Fog-computing-based radio access networks: issues and challenges. *IEEE Network, 2016.*

[30] Z. Sanaei et al. Sami: Service-based arbitrated multi-tier infrastructure for mobile cloud computing. In *ICCC*. IEEE, 2012.

[31] J. K. Sankaran. On a dynamic auction mechanism for a bilateral assignment problem. *Mathematical Social Sciences*, 1994.

[32] M. Satyanarayanan et al. Edge analytics in the internet of things. *IEEE Pervasive Computing, 2015.*

[33] M. Satyanarayanan et al. The case for VM-based cloudlets in mobile computing. *Pervasive Computing*, 2009.

[34] L. S. Shapley et al. The assignment game I: The core. *International Journal of Game Theory*, 1971.

[35] W. Shi et al. An online auction framework for dynamic resource provisioning in cloud computing. *ACM SIGMETRICS, 2014.*

[36] S. Uppoor et al. Generation and analysis of a large-scale urban vehicular mobility dataset. *Transactions on Mobile Computing*, 2014.

[37] L. Yang et al. A framework for partitioning and execution of data stream applications in mobile cloud computing. *SIGMETRICS*, 2013.

[38] B. Zhang et al. The cloud is not enough: Saving IoT from the cloud. In *HotCloud, 2015.*