Self-Tuning Service Provisioning for Decentralized Cloud Applications

Raul Landa, Marinos Charalambides, Richard G. Clegg, David Griffin, and Miguel Rio

***Abstract*—Cloud computing has revolutionized service delivery by providing on-demand invocation and elasticity. To reap these benefits, computation has been displaced from client devices and into data centers. This partial centralization is undesirable for applications that have stringent locality requirements, for example low latency. This problem could be addressed with large numbers of smaller cloud resources closer to users. However, as cloud computing diffuses from within data centers and into the network, there will be a need for cloud resource allocation algorithms that operate on resource-constrained computational units that serve localized subsets of customers. In this paper we present a mechanism for service provisioning in distributed clouds where applications compete for resources. The mechanism operates by enabling execution zones to assign resources based on Vickrey auctions, and provides high-quality probabilistic models that applications can use to predict the outcomes of such auctions. This allows applications to use knowledge of the locality distribution of their clients to accurately select the number of bids to be sent to each execution zone and their value. The proposed mechanism is highly scalable, efficient and validated by extensive simulations.**

***Index Terms*—Cloud Resource Management, Decentralized Cloud Applications, Vickrey Auctions, Quality of Experience.**

I. Introduction

Cloud computing has been very successful, delivering applications and services in a scalable manner through on-demand computation. It has allowed applications to elastically cope with changing demand for computing resources, and to exploit economies of scale in multi-tenancy data centers. However, this has meant that in many cases execution points can only be selected from a limited number of options; this could potentially reduce the quality of experience (QoE) provided to the customers, or increase the bandwith requirements between geographically dispersed customers and cloud data centers.

Many resource-demanding services (including personalized real-time video, games or processing of high bandwidth streams for security, safety and health-care monitoring) are not suited to being centralized in a relatively small number of data centers where higher network delays and low throughput can have a serious impact on the QoE experienced by many users. Application providers do not normally have the means to position cloud application logic very close to their users in order to meet tight quality constraints or to avoid congested or expensive network paths for high bandwidth flows.

There is a current trend for deploying cloud resources closer to users as telcos and ISPs deploy data centers at the network edge. In addition, routers and other network equipment can now also perform general-purpose computation for virtualized network functions; these can be deployed by ISPs themselves or offered to third party providers to deploy applications and services [1]. Such a *distributed cloud* drastically reduces RTT to customers and greatly relaxes the end-to-end bottleneck bandwidth requirements between end users and these localized cloud points of presence. Distributed cloud resources can be deployed in various places throughout the Internet: in network-edge points of presence close to users, as an extension to typical CDNs; in specialized data centers owned and operated by ISPs; and in traditional data centers and service farms operated by cloud providers. Unfortunately, each one of these small cloud execution points will be unable to provide the essentially boundless elasticity that purpose-built data centers can. This means that, in many instances, there can be more requests for resources at a given execution point than there are resources available. In these cases, resource allocation mechanisms that deal with service contention will be required.

In this paper we present a resource allocation mechanism for service provisioning in distributed clouds where computation resources are made available much closer to users, possibly within routers themselves. This will save bandwidth, provide better user QoE and potentially comply with regulatory frameworks that demand services to run in specific regions/countries/states. Applications can then use these resources to deploy demanding services. Of course, when securing resources, applications will need to balance the benefit that these resources bring to the QoE of their users with the costs that they need to pay to the computation resource providers. To this end, we propose an auction-based resource allocation approach that allows cloud providers to provide computation capacity in a decentralized manner, while at the same time allowing applications to implement effective cost-benefit tradeoffs. We take as given the well-known properties of the Vickrey auction (eg. *truthful revelation* and *ex-post efficiency* [2]) and focus on their use as a vehicle to implement a self-adaptive mechanism for the dynamic allocation of service slots, obviating the need of a logically centralized clearinghouse.

In the paper we show that the proposed approach provides an effective option to deal with contention and oversubscription in decentralized cloud scenarios. We prove that it is both scalable and efficient, but also immune to false-name attacks. We present both the analytical work that underpins our design and an extensive set of simulations that demonstrate its behavior in many cases of interest. The dynamic behavior of the algorithm has been investigated using three different convergence approaches and it is shown that the algorithm

The authors are with the Department of Electronic and Electrical Engineering, University College London, UK.

converges to the same solutions.

The paper is organized as follows. In Section II we describe the main actors of our proposed mechanism. We continue in Section III by explaining the analytical foundations of our resource allocation mechanism. In Section IV we present the results of evaluating our proposed mechanisms through simulation in a wide variety of situations. Finally, we present an overview of the related work in auction-based resource allocation in Section V and our conclusions in Section VI.

II. System Overview

In our proposed system we assume the existence of a large number of *execution zones* (EZ), where computational capabilities are offered to applications. Resources in each EZ are managed by a *zone manager* (ZM), which is operated by the infrastructure provider. Applications can deploy *services* (i.e. self-contained application logic components) on these EZs under the direction of an *application manager* (AM), which is operated by the application provider and implements the appropriate cost-benefit tradeoffs for that application. The AMs belonging to various applications then compete for resources on the EZs. Rather than relying on a centralized resource allocator to grant AMs access to EZ resources, we allow each EZ to individually perform resource allocation decisions. There is a rich literature on how to solve this problem [3]–[5]; we propose the use of a market system to allow EZs to allocate resources to those applications that value them the most (see [6]–[9]). We take advantage of the well known properties of Vickrey auctions [2] to allocate resources to applications (see [10]–[15] for other examples of auctions used for resource allocation in grid, peer-to-peer and cloud computing). Our mechanism allows each AM to bid for resources on the desired EZs according to its individual policies, and to implement its own tradeoff between user satisfaction and infrastructure cost. This involves considering the monetary cost of the resources together with the cost of users being blocked due to resource unavailablity.

We consider latency- and bandwidth-sensitive services which require accurate placement of service instances so that the network performance metrics between users and cloud-based services are within acceptable bounds to ensure QoE. There are two dimensions to QoE in our model. Firstly there is the quality experienced by the users of an application in terms of the latency, bandwidth and response time of a session. This is determined by an AM selecting EZs that are topologically close to the users in terms of network metrics, such as latency, and by the EZ offering sufficient computational capabilities (e.g. CPU, memory) for the application. This aspect of QoE is determined by decisions made by the AM to map user demand to appropriate EZs and is part of the AM logic outside of the auction mechanism described in this paper. The outcome of this decision determines the demand on a specific EZ and hence will drive the quantity of bids an AM should make to a specific EZ. The second dimension of QoE, termed *QoS cost*, is related to the blocking probability of user session requests at an EZ. Our model deals with both dimensions of QoE, but only the blocking probability is directly impacted by the auction outcome. Latency, throughput and other metrics of an active session are indirectly covered by the prior selection of the EZs where an AM will place bids for service slots. For this reason, and in order to maintain our architectural decoupling between AM and EZ functions, we assume that bids from AMs are relevant only for one specific EZ. Off-loading of demand from one EZ to another may violate QoE constraints and the placement decisions made by the AMs. Of course, a cloud provider with multiple physical data centers within a region may act as a single EZ if the underlying locations are equivalent in terms of latency, bandwidth and other network performance metrics.

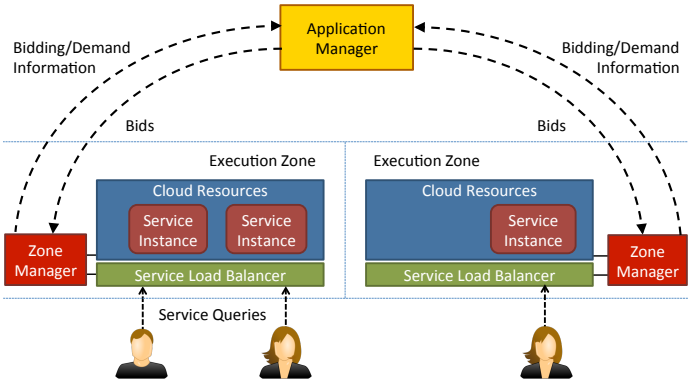


Fig. 1. Notional architecture

We focus our attention on the dynamic use of EZ resources by AM. To simplify the resource allocation tasks of EZs, we will assume that each EZ will offer a discrete number of *identical service slots*. Rather than expressing service execution requirements in terms of CPU speed, amount of memory, etc. EZs abstract the underlying computational resources as service slots. Cloud providers, such as Amazon EC2, offer multiple virtual computing instance types that vary in terms of the processing and storage capabilities of the underlying hardware resources. We assume that an AM has identified the resource type offered by a data center that is best suited to their applications, e.g. to optimize compute power, storage or memory, or the use of specialized resources such as GPUs. A service slot for an application will be mapped to a single instance type and an AM will therefore bid on only one type of instance. Our model assumes that a cloud provider offering multiple instance types could operate multiple, collocated EZs, one for each instance type and therefore a separate auction where AMs offering applications with similar requirements will compete for resources. We assume that prior negotiation has established the type and quantity of resources required to support a service slot for the applications being deployed by the AM. In addition to its scalability benefits (see Section IV-C), this abstraction simplifies the resource modeling in EZs and allows a more straightforward calculation of the tradeoffs involved. Our model therefore allows heterogeneous application requirements and heterogeneous resources to be accommodated without introducing the complexity of a multi-parameter auction.

According to the definitions above, our proposed mechanism, which is depicted in Fig. 1, can be explained as follows.

AMs determine which EZs are capable on computational grounds for hosting their application. From this subset of EZs an AM will determine how demand from their users should be mapped to EZs so that network metrics, such as latency and bandwidth, do not violate application QoE requirements. Users will typically be distributed over a wide area and a single EZ is unlikely to be positioned to achieve the required QoE to all user locations. Hence, our model assumes that an AM will select multiple EZs to deploy the application and will bid for resources in all EZs it has selected (the quantity of service slots being determined at each EZ by the user demand in that location). Application users generate *service queries*, which are directed to their local EZs by existing methods like differentiated DNS resolution and HTTP redirection, as configured by the application provider. These queries are received by ZMs and mapped to the service slots that the application has available on that zone. Each AM can secure a number of service slots for its application in each EZ; this is achieved by sending bids to the ZMs specifying the number of service slots desired and the bid amount that they are willing to pay per service slot. The ZM then allocates its available service slots by running a multi-item Vickrey auction [16]; the top bidding AM bids are hence selected. AMs make a centralized decision on which EZs should host their application to meet QoE requirements and on the quantity of resources required to meet the demand at that EZ (see Fig. 1). Each EZ runs an auction for resources with bids being generated by the set of AMs that have selected the EZ. There is a set of independent auctions, one at each EZ (see Fig. 3), where each auction is for local resources only and therefore no coordination is required between independent auctions.

Among alternatives, eg. open ascending / descending bid auctions, we have chosen to use Vickrey auctions due to their low overhead, which preserves system scalability. Furthermore, compared to other sealed bid auction types, eg. first price, in Vickrey auctions, true value bidding is the dominant strategy (*truthful revelation*). Since the Vickrey auction is a VCG mechanism [2], it is *ex-post efficient*: it allocates resources in a way that is optimal for the entire system. This is the reason it is frequently used as a protocol component in self-organising resource allocation [2] [17].

III. Auction-Based Resource Allocation for the Decentralized Cloud

We assume that the AM can determine, by external means, the total user demand associated with a given EZ. Then, for each AM the resource allocation problem becomes: given the bidding profile at every EZ, for how many service resource units should the AM bid, and at what price? Formally, each AM b_j must determine the number of service slots m_{ij} it will bid for at each EZ j,as well as the bid price v_{ij} it must offer, according to the estimated user demand and expected service quality.

We now present our analytic model of the EZ, and how it implements resource allocation using Vickrey auctions. In a standard Vickrey second price auction the bidders bid for a single item. The highest bid wins and pays the price of the second highest bid. This can be shown (under certain assumptions) to mean that rational bidders will bid according to their true valuation for the item. In a multiunit Vickrey auction with identical items, the bidders bid for K items and may each bid multiple times. The highest K bids win and they pay the combined value of the next K bids (the first K losing bids). This preserves the property that rational bidders bid according to how much they value the item.

In the context of this paper, we assume that each EZ n_i can offer up to M_i service slots (see Table I), and that each AM b_j sends m_{ij} bids to n_i, each at a value v_{ij}. The EZ receives these bids and ranks them in increasing order of value, and the top M_i win a service slot. Let X^{ij} be the number of bids that AM b_j wins in EZ n_i. According to the definition of a Vickrey auction, the payment that will be levied on manager b_j is the sum of the top X^{ij} losing bids.

By the construction of a Vickrey auction the value of the bids that n_i makes in b_j are a true reflection of the benefit it would gain from those bids if they were successful. Therefore, the utility U_{ij} that b_j obtains from this interaction with n_i is just the difference between the sum of the values of its X^{ij} winning bids and the sum of the top X^{ij} losing bids (its payment to the EZ). The outcome of this process is that each AM calculates the price of its bids to each EZ according to the value it expects to derive from service slots located in that EZ.

A. Resource Allocation Properties

Even though the Vickrey auction has interesting advantages (e.g. *incentive compatibility* and *ex-post efficiency*), it is not revenue-maximizing [2] [17]. In fact, for uncontested Vickrey auctions, the bidders can obtain items at zero cost. For the present paper we assume that AMs pay a monthly fixed fee to gain access to the EZ infrastructure; this would allow them to use uncontested resources. If AMs wish to gain access to EZ resources for which there is contention, they will need to submit nonzero bids and pay for these bids in addition to the fixed flat fee. In contrast to other contributions in the literature, we assume that bids are made in terms of actual currency and the EZs receive income from both the monthly rental fees and non-zero winning bids from AMs.

We base our auction mechanism on two simplifying assumptions. First, we will use the *independent private values* model in the sale of M_i service slots to N_i managers. This means that the value that each AM b_j gives to service slots auctioned by an EZ n_i is private, and is not a function of the values that other AMs give to service slots offered by n_i. This simplifies the valuation problem, decoupling the decisions for each AM. Second, we will assume that AMs experience *no synergy* in acquiring increasing numbers of service slots. This means that for each manager b_j, the value of obtaining X^{ij} service slots from n_i is just X^{ij} times the value of obtaining a single service slot from n_i. Therefore, there are no complementarity or substitution effects between different service slots, and all service slots from a given EZ have the same value.

In addition to making the problem mathematically tractable, these two assumptions have an additional benefit: they make

our proposed mechanism *false-name proof* [18] [19]. In a combinatorial auction bidders bid for sets of non-identical items, and they have independent valuations for each one of these sets. This makes the system vulnerable to *false-name* attcks, in which bidders can use additional identities to submit strategic bids that allow them to obtain items at a reduced cost. Although false-name proofness is in general impossible using Vickrey auctions [18], our proposed mechanism avoids this problem by mandating service slots within each EZ to be identical (for combinatorial auctions, different auction mechanisms can regain the false-name-proof property [19]).

Theorem 1. *In a system where service slots are allocated to AMs following the mechanism described in Section III-A, there is no incentive for AMs to submit bids at any valuation other than their true valuation.*

Proof. The false-name attack is geared towards obtaining the same set of items in the auction but at a lower cost; hence, our analysis assumes that the bidder will only bid for its desired number of service slots. Further, since bidders can only specify the number of slots that they bid for and the unitary valuation that they have for each one, we only need to consider three possibilities: that the bid is equal, lower or higher than their true value.

1) **False-name bids are made at the same valuation as true-name bids.** This case is indistinguishable from simply submitting the same number of bids using a single name. Hence, in this case the bidder has no incentive to use an alternate identity.
2) **False-name bids are made at a lower valuation than true-name bids.** We can distinguish two cases here. If false-name bids are low enough so that the bidder fails to secure service slots that it would have secured by revealing its true valuation, the bidder will be forced to forgo service slots which were available at a price that the bidder was willing to pay. Consequently, it will have experienced a lower utility than it would have had it bid its true valuation. On the other hand, if the false-bids are lower than its true valuation, but still high enough to secure all required service slots, the amount paid is identical to that obtained by bidding normally using its true name. Hence, *under-bidding* with an alternative identity can not increase its utility, and may potentially decrease it.
3) **False-name bids are made at a higher valuation than true-name bids.** Again, we can distinguish two cases. If false-name bids are not high enough so that the bidder wins service slots that it would not have won by bidding its true valuation, its utility is unchanged. If false-name bids are high enough to make the bidder win service slots that it would not have won by bidding its true valuation, its utility will be negative for those service slots. To see why, consider that the true valuation for those service slots is necessarily lower than that of the lowest bid in the winning set (by the definition of our auction mechanism in Section III-B). When these bids displace a number of previously winning bids, these will become the highest losing bids, and hence, will define the price that the bidder will have to pay. However, this price is higher than its true valuation, and hence, will lead to a negative utility. It follows that *over-bidding* with an alternative identity can not increase its utility, and may potentially decrease it.

□

B. Estimating the Expected Number of Successful Bids

To allow an AM to estimate the expected number of bids it will win and the total price it will pay when placing a given number of bids at a given price, each with a given ZM, it needs to estimate the level of competition it will face in the auction from other AMs. Formally, this is achieved by determining the number of competing bids it will encounter. Thus, rather than the total number of bids N_i received by n_i, an AM b_j is interested in the number of bids with which its bids must compete. This involves subtracting from N_i those bids sent to n_i by b_j itself. We shall denote this number of competing bids as N_i^j, the *total competing load* of manager b_j at EZ n_i. By definition, $N_i^j = \sum_{k \neq i} m_{kj}$.

We now define two basic load-related states for the EZ. We call the case where $M_i > N_i^j$ the *surplus* state, and the case where $M_i \leq N_i^j$ as the *scarcity* state. In the surplus state there are more service slots in auction than bids submitted by competing AMs, and b_j can expect to have some of its bids accepted at zero price. Winning bids pay nonzero prices only in the scarcity state, when b_j competes with other managers and there are losing bids.

It will be advantageous for AMs to have a model of the expected auction outcome after bidding with a given EZ. To this end, we investigate how an AM b_j can predict, given a hypothetical bidding value v_{ij}, the expected number of bids it will win when submitting bids to a particular ZM n_i. Since by the *truthful revelation* property of the Vickrey auction [2], [17] bidders have no incentive to bid strategically, we can safely assume that v_{ij} is the true value that every AM estimates for service slots being auctioned by n_i. In what follows, we shall refer to a given execution manager n_i or its associated EZ interchangeably; likewise, we will denote as b_j both the AM and the application itself.

We simplify our analysis by modeling the bid values as a random variable V^i with a probability density f^{V^i} and cumulative distribution F^{V^i}. Then, $F^{V^i}(v_b) - F^{V^i}(v_a) = \int_{v_a}^{v_b} f^{V^i}(\omega) d\omega$ gives the proportion of values between v_a and v_b for all those bids that were sent to n_i from all AMs. Since AMs do not bid strategically, EZs can compile these value probability distributions simply by performing elementary frequency counts (histograms) over their received bids.

Again focusing on the prospective competition for AM b_j, in the model presented here we treat the N_i^j competing bids that n_i receives in a given interval T as a set of N_i^j realizations of the random variable V^i. We denote this vector of variates drawn from f^{V^i} as $V^i = \{V^i{}_1, V^i{}_2, \ldots, V^i{}_{N_i^j - 1}, V^i{}_{N_i^j}\}$. To model a bidding round in a Vickrey auction, we assume that the EZ n_i constructs V^i using the bid values it receives for that round. Then, it sorts V^i in ascending order, creating $\bar{V}^{ij} =$

TABLE I
AUCTION-BASED RESOURCE ALLOCATION MODEL NOTATION

N_i	Total number of bids that zone manager n_i receives
N_i^j	Total number of bids that zone manager n_i receives which compete with those placed by b_j
M_i	Total number of service slots offered in execution zone n_i
m_{ij}	Total number of bids that application manager b_j sends to zone manager n_i
v_{ij}	Monetary value for each bid that application manager b_j sends to to zone manager n_i
X^{ij}	Number of bids that b_j wins on n_i
X_{free}^{ij}	Number of contention free service slots (ie. obtainable at zero auction cost) that b_j can obtain when submitting bids to n_i
$X_{\max}^{ij}$	Maximum number of bids that b_j can win by bidding with n_i
$f^{V^i}(v_{ij})$	Probability density function of bid values received by auctioneer n_i
$F^{V^i}(v_{ij})$	Cumulative distribution function of bid values received by auctioneer n_i
$f_{(k)}^{V^i}(v_{ij})$	Probability density function of the k-th order statistic of the bid values received by auctioneer n_i
$F_{(k)}^{V^i}(v_{ij})$	Cumulative distribution function of the k-th order statistic of the bid values received by auctioneer n_i
ϕ_{ij}	Maximum losing rank in the Vickrey auction; bidding round after which bids sent to n_i by b_j will necessarily be lost
μ_{ij}	Expected number of successful bids that b_j will have when bidding with n_i
$\mathrm{E}[C_{ij}]$	Expected monetary cost that b_j will incur when submitting m_{ij} bids to n_i at a bid price v_{ij}
$\mathrm{E}[Q_{ij}]$	Expected QoS cost that b_j will incur when submitting m_{ij} bids to n_i at a bid price v_{ij}
$\mathrm{E}[T_{ij}]$	Expected total cost (monetary and QoS) that b_j will incur when submitting m_{ij} bids to n_i at a bid price v_{ij}

$\{V^i{}_{(1)}, V^i{}_{(2)}, \ldots, V^i{}_{(N_i^j-1)}, V^i{}_{(N_i^j)}\}$, keeps the top M_i bids in $\bar{V}^{ij}$, and discards the rest.

Theorem 2. *Assume a Vickrey auction where application managers bid for execution slots within zones. Let v_{ij} be the value to application manager b_j of winning one slot in zone n_i and assume that the value of winning s such slots is sv_{ij}. Let $F^{V^i}(v_{ij})$ be the CDF of the bid values received by n_i from b_j. Let N_i^j be the total number of bids received by n_i excluding those placed by b_j. Let ϕ_{ij} be the maximum losing rank in the auction and let $X_{\max}^{ij}$ be the maximum number of bids that b_j can win by bidding with n_i. The expected number of slots won by b_j in zone n_i is given by*

$$\mu_{ij} = \sum_{k=1}^{X_{\max}^{ij}} I(F^{V^i}(v_{ij})\,;\, \phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)., \quad (1)$$

where $I(x;a,b)$ is the regularized incomplete beta function.

Proof. We are interested in the probabilities of the various possible outcomes of the auction (see Figure 2). We consider a situation where every AM in the system but b_j has already submitted all their bids to n_i, so that the N_i^j, M_i and m_{ij} are set. In this case, since N_i^j is the total number of bids that n_i will receive in addition to the m_{ij} that b_j will send, we have that $N_i = N_i^j + m_{ij}$. If $N_i^j \geq M_i$, n_i is in the scarcity state, and bids from b_j will face competition. In particular, we see that the lowest bid that b_j will have to surpass will be the $(N_i^j - M_i + 1)$-th highest bid received by n_i, which is the *last winning bid in its absence*.

On the other hand, if $M_i > N_i^j$, n_i is in the surplus state and b_j will have access to $X_{\text{free}}^{ij} = M_i - N_i^j$ contention-free service slots: the first X_{free}^{ij} bids from b_j will win without competition and have zero cost. Any bids following, however, will face the same competition situation detailed above, although in this case with a reduced effective number of bids $m'_{ij} = m_{ij} - X_{\text{free}}^{ij} = m_{ij} - M_i + N_i^j$. Thus, we see that if $m_{ij} < X_{\text{free}}^{ij}$, the AM wins all its bids at zero cost, and if $m_{ij} \geq X_{\text{free}}^{ij}$, X_{free}^{ij} of its bids win at zero cost and m'_{ij} face the EZ in the scarcity state. Without loss of generality, we focus on the scarcity state and assume that $N_i^j > M_i$.

We see in Figure 2 that b_j will win no bids with a probability $\mathbb{P}[v_{ij} < V^i{}_{(N_i^j - M_i + 1)}]$, and it will win exactly one bid with probability $\mathbb{P}[V^i{}_{(N_i^j-M_i+1)} < v_{ij} < V^i{}_{(N_i^j-M_i+2)}]$. This pattern continues until either all the bids that were submitted to the EZ have been considered (the case shown in Figure 2) or all the service slots that n_i has in auction have been considered. This bounds X^{ij}, the maximum number of bids that an AM can win, to either the total number of units in auction M_i or the number of bids that b_j submits, m_{ij}.

Thus, we have that $X_{\max}^{ij} = \min(m_{ij}, M_i)$, and this can happen with a probability of $\mathbb{P}[V^i_{(N_i^j - M_i + X_{\max}^{ij})} < v_{ij}]$. For μ_{ij}, the expected number of successful bids that b_j will have per auction, we have that

$$\mu_{ij} = \mathbb{E}[X^{ij}] = \sum_{k=1}^{X_{\max}^{ij}-1} k\,\mathbb{P}[V^i_{(\phi_{ij}+k)} < v_{ij} < V^i_{(\phi_{ij}+k+1)}] + X_{\max}^{ij}\mathbb{P}[V^i_{(\phi_{ij}+X_{\max}^{ij})} < v_{ij}], \quad (2)$$

where $\phi_{ij} = N_i^j - M_i$. Since we have that $\mathbb{P}[V^i{}_{(\phi_{ij}+k)} < v_{ij} < V^i{}_{(\phi_{ij}+k+1)}] = \mathbb{P}[V^i{}_{(\phi_{ij}+k)} < v_{ij}] - \mathbb{P}[V^i{}_{(\phi_{ij}+k+1)} < v_{ij}]$, we see that the expression for μ_{ij} telescopes and can be trivially simplified. Thus, it follows that

$$\mu_{ij} = \mathbb{E}[X^{ij}] = \sum_{k=1}^{X_{\max}^{ij}} \mathbb{P}[V^i{}_{(\phi_{ij}+k)} < v_{ij}]. \quad (3)$$

We find $\mathbb{P}[V^i{}_{(\phi_{ij}+k)} < v_{ij}]$ by noting that it corresponds to the $(\phi_{ij}+k)$-th *order statistic* of the probability density f^{V^i} of bid values [20], [21]. Formally, order statistics can be defined as follows.

Definition 1. *Let $V^i \geq 0$ denote a continuous random variable with probability density $f^{V^i}(v)$, and cumulative distribution $F^{V^i}(v) = \int_0^v f^{V^i}(\omega)d\omega$. Let $(V^i_{(1)}, V^i_{(2)}, \ldots, V^i_{(N_i^j)})$ denote a random sample of size N_i^j drawn on V^i, and ordered so that $V^i_{(1)} < V^i_{(2)} < \ldots < V^i_{(N_i^j)}$. Then, the $V^i_{(1)}, V^i_{(2)}, \ldots, V^i_{(N_i^j)}$ are collectively known as the* order statistics *derived from V^i. The probability density for the k-th order statistic will be denoted as $f^{V_i}_{(k)}(v)$, and can be calculated [21] as*

$$f^{V_i}_{(k)}(v) = \frac{N_i^j! F^{V^i}(v)^{k-1}(1-F^{V^i}(v))^{N_i^j-k} f^{V^i}(v)}{(k-1)!(N_i^j-k)!}. \quad (4)$$

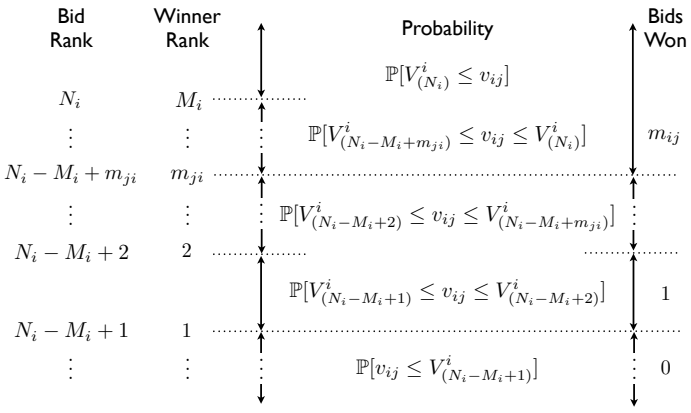


Fig. 2. Calculating the expected number of won bids

We denote the cumulative distribution of the $(\phi_{ij}+k)$-th order statistic of V^i, as $F^{V^i}_{(\phi_{ij}+k)}(v_{ij})$, so that

$$\mathbb{P}[{V^i}_{(\phi_{ij}+k)} \le v_{ij}] = F^{V^i}_{(\phi_{ij}+k)}(v_{ij}) = \int_0^{v_{ij}} f^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega, \tag{5}$$

where $f^{V^i}_{(\phi_{ij}+k)}(\omega)$ denotes the probability density for the $(\phi_{ij}+k)$-th value in our sorted vector $\bar{V}^{ij}$, and $F^{V^i}_{(\phi_{ij}+k)}(v_{ij})$ denotes its cumulative distribution. By substituting the definition for $f^{V^i}_{(\phi_{ij}+k)}(\omega)$ in (5) and simplifying, we find that

$$F^{V^i}_{(\phi_{ij}+k)}(v_{ij}) = \frac{B(F^{V^i}(v_{ij})\,;\,\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)}{B(\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)},$$

where $F^{V^i}(v_{ij})$ is the cumulative density of the bid values submitted to n_i valuated at v_{ij}, and $B(\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)$ and $B(x\,;\,\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)$ are, respectively, the *beta function* $B(a,b)$ and the *incomplete beta function* $B(x\,;\,a,b)$ with parameters $a=\phi_{ij}+k$ and $b=N_i^j-(\phi_{ij}+k)+1$. Formally, these are defined as

$$B(x\,;\,a,b) = \int_0^x \omega^{a-1}(1-\omega)^{b-1}d\omega \quad [0\le x\le 1]$$

$$B(a,b) = B(1\,;\,a,b) = \int_0^1 \omega^{a-1}(1-\omega)^{b-1}d\omega.$$

We can further simplify (5) by using $I(x\,;\,a,b)$, the *regularized incomplete beta function* with parameters a and b, which is defined as

$$I(x\,;\,a,b) = \frac{B(x\,;\,a,b)}{B(a,b)} \quad [0\le x\le 1].$$

We can see then that

$$\mathbb{P}[{V^i}_{(\phi_{ij}+k)} \le v_{ij}] = I(F^{V^i}(v_{ij})\,;\,\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1),$$

where $I(x\,;\,\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1)$ is $I(x\,;\,a,b)$ with parameters $a=\phi_{ij}+k$ and $b=N_i^j-(\phi_{ij}+k)+1$. By substituting this last expression in (3), we then have (1) from the theorem.

$$\mu_{ij} = \sum_{k=1}^{X^{ij}_{\max}} I(F^{V^i}(v_{ij})\,;\,\phi_{ij}+k, N_i^j-(\phi_{ij}+k)+1).$$

Since $I(0;a,b)=0$ and $I(1\,;\,a,b)=1$, we have that $\lim_{v_{ij}\to 0}\mu_{ij}=0$ and $\lim_{v_{ij}\to\infty}\mu_{ij}=X^{ij}_{\max}$, as expected: in the first case no bids are won, while in the second one the maximum number of feasible service slots are won. This completes the proof. □

C. The Expected Resource Cost

We now calculate the expected cost of an AM when bidding at a given EZ.

Theorem 3. *Given the auction mechanism from theorem 2 the expected cost paid by an AM b_j to a ZM n_i is given by*

$$E\,[C_{ij}] = v_{ij}\mu_{ij} - \int_0^{v_{ij}} \mu_{ij}(\omega)d\omega. \tag{6}$$

Proof. This cost, for each won bid, equals the price that b_j needs to pay to n_i. Then, for the cost C_{ij} paid by b_j if it sends m_{ij} bids to n_i, we have for $N \le X^{ij}_{\max}$ won bids

$$C_{ij} = \sum_{k=1}^{N} V^i_{(\phi_{ij}+k)} \quad \text{if} \quad V^i_{(\phi_{ij}+N)} < v_{ij}$$

and thus we have for the expected cost $\mathbb{E}[C_{ij}]$ that

$$\mathbb{E}[C_{ij}] = \sum_{k=1}^{X^{ij}_{\max}} \int_0^{v_{ij}} \omega f^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega,$$

where, again, $f^{V^i}_{(\phi_{ij}+k)}(\omega)$ is the probability density of the $(\phi_{ij}+k)$-th order statistic of V^i. The integral within the summation can be partially evaluated and simplified using integration by parts, and we find that

$$\int_0^{v_{ij}} \omega f^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega = \omega F^{V^i}_{(\phi_{ij}+k)}(\omega)\Big|_0^{v_{ij}} - \int_0^{v_{ij}} F^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega.$$

Hence, we have that

$$\mathbb{E}[C_{ij}] = \sum_{k=1}^{X^{ij}_{\max}} \left[v_{ij}F^{V^i}_{(\phi_{ij}+k)}(v_{ij}) - \int_0^{v_{ij}} F^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega \right],$$

and therefore, if we exchange summation and integration, that

$$\mathbb{E}[C_{ij}] = v_{ij}\sum_{k=1}^{X^{ij}_{\max}} F^{V^i}_{(\phi_{ij}+k)}(v_{ij}) - \int_0^{v_{ij}} \sum_{k=1}^{X^{ij}_{\max}} F^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega.$$

Let $\mu_{ij}(\cdot)$ represent μ_{ij} expressed as a function of the bid value v_{ij} and hence

$$\begin{aligned}\mathrm{E}\,[C_{ij}] &= v_{ij}\mu_{ij} - \int_0^{v_{ij}} \sum_{k=1}^{X^{ij}_{\max}} F^{V^i}_{(\phi_{ij}+k)}(\omega)d\omega \\ &= v_{ij}\mu_{ij} - \int_0^{v_{ij}} \mu_{ij}(\omega)d\omega.\end{aligned}$$

□

D. The Expected QoS Cost

In this section we consider the QoS cost that an AM expects. The QoS cost can be considered as a penalty to QoS that the application has, given the number of service slots secured.

Theorem 4. *Given the auction mechanism from theorem 2 define Q_{ij} as the QoS cost that users of application b_j experience in EZ n_i. This is a function of the number of service slots secured. We define $Q_{ij}(k)$ as the QoS cost from securing exactly k slots in n_i. Then we have*

$$\mathbb{E}[Q_{ij}] = Q_{ij}(0) + \sum_{k=1}^{X_{\max}^{ij}} \Delta Q_{ij}(k) \qquad (7)$$
$$I(F^{V^i}(v_{ij})\,;\, \phi_{ij}+k, N_i^j - (\phi_{ij}+k)+1),$$

where $\Delta Q_{ij}(k) = Q_{ij}(k) - Q_{ij}(k-1)$ is the QoS cost difference achieved by adding the kth service slot.

Proof. In this case, we assume that customer queries will receive a QoS that is a function of the current number of service slots that an application has obtained in a given EZ. If we define $Q_{ij}(k)$ as the QoS cost that users of application b_j experience when using EZ n_i (with k service slots secured by b_j), we can calculate the expected QoS cost as

$$\begin{aligned}\mu_{ij}^Q = \mathbb{E}[Q_{ij}] &= Q_{ij}(0)\,\mathbb{P}[v_{ij} < V^i_{(\phi_{ij})}] \\ &+ \sum_{k=1}^{X_{\max}^{ij}-1} Q_{ij}(k)\,\mathbb{P}[V^i_{(\phi_{ij}+k)} < v_{ij} < V^i_{(\phi_{ij}+k+1)}] \\ &+ Q_{ij}(X_{\max}^{ij})\,\mathbb{P}[V^i_{(\phi_{ij}+X_{\max}^{ij})} < v_{ij}].\end{aligned}$$

By following a reasoning identical to (3), we see that

$$\mu_{ij}^Q = Q_{ij}(0) + \sum_{k=1}^{X_{\max}^{ij}} \Delta Q_{ij}(k)\mathbb{P}[V^i{}_{(\phi_{ij}+k)} < v_{ij}].$$

The theorem follows immediately. □

By comparing (1) and (7) we see that whereas k runs from 1 in (1), it runs from 0 in (7). The reason for this is that, whereas $k = 0$ in (1) contributes nothing to μ_{ij}, $k = 0$ in (7) has an important contribution to μ_{ij}^Q because it represents the situation where no client requests are served, and hence, the maximum QoS cost that can be incurred.

To understand the boundary properties of μ_{ij}^Q, we assume that the QoS cost $Q_{ij}(k)$ decreases as the number of service slots k (bids won) increases. This implies that $\Delta Q_{ij}(k)$ will be negative for all values of k. Hence, μ_{ij}^Q attains its maximal value of $Q_{ij}(0)$ for $k = 0$ and it then decreases with increasing k. In addition, we have that $\lim_{v_{ij}\to 0} \mu_{ij}^Q = Q_{ij}(0)$ and $\lim_{v_{ij}\to\infty} \mu_{ij}^q = Q_{ij}(X_{\max}^{ij})$, as expected.

E. Bid Determination: The Expected Total Cost

Suppose that an AM b_j wishes to determine the number and value of bids that it must submit to a given execution manager n_i, and it wishes to do so taking into account both the resource cost and client QoS. This can be achieved by selecting the combination of m_{ij} and v_{ij} that minimize a *total cost* measure that includes both QoS and monetary bidding effects. We use the simplest possible definition, in which the *total expected cost* is simply the sum of the resource and the QoS costs. However, since the resource cost is in monetary units and the QoS cost is in QoS units, we require a mapping between these two units of measurement so that these costs can be added. We assume a linear mapping based on a static coefficient ζ that gives the monetary cost associated with a given QoS cost. This does not impose any loss in generality, as the nonlinear aspects of QoS cost will be considered separately: the role of ζ is only to allow the joint consideration of QoS and monetary costs into a single model. Then, for this *total cost* T_{ij} we have

$$\mathrm{E}\left[T_{ij}\right] = \mathrm{E}\left[C_{ij}\right] + \zeta \mathrm{E}\left[Q_{ij}\right]. \qquad (8)$$

However, in order for T_{ij} to be fully defined, it is necessary to define the quality cost $Q_{ij}(k)$. To achieve this, we assume that each EZ i will provide service to a given population of customers that generate service requests at a rate of λ_{ij} per unit time following a Poisson distribution. Further, we assume that each service slot can process service requests at a rate of r_i per unit time, with the service time being exponentially distributed. Finally, we assume that each service slot can only process one request at a time. The queueing model, therefore has exponential interarrival times, exponential service times, N servers and at most N customers. In standard Kendall notation [22], this is an M/M/N/N model.

We use the loss probability of the the M/M/N/N model to define the QoS cost $Q_{ij}(k)$. Intuitively, $Q_{ij}(k)$ is defined as the probability that a customer request will be denied service (i.e. blocked) from EZ n_i, given a service request rate of λ_{ij} and a service completion rate of r_i. Formally, if we define the offered load $\rho_{ij} = \frac{\lambda_{ij}}{r_i}$ and the full-blocking cost B_{ij}, we have

$$Q_{ij}(k) = \frac{\frac{\rho_{ij}^k}{k!}}{\sum_{n=0}^{k} \frac{\rho_{ij}^n}{n!}} B_{ij}. \qquad (9)$$

By using the recursive definition of the Erlang B loss probability [22], we find that

$$\Delta Q_{ij}(k) = \left[1 - Q_{ij}(k-1) + \frac{k}{\rho_i}\right] Q_{ij}(k), \qquad (10)$$

where the recursion terminates with $Q_{ij}(0) = 1$. This will allow our simulation engine to calculate $Q_{ij}(k)$ efficiently.

With these definitions, we can formulate the bid planning problem for each AM as

$$\underset{m_{ij}\in\mathbb{N}_{\geq 0}, v_{ij}\in\mathbb{R}_{\geq 0}}{\text{Minimize:}} \quad \mathrm{E}\left[T_{ij}\right]. \qquad (11)$$

This problem could be solved by using nonlinear mixed integer programming. In the case of this paper, however, the state space is sufficiently small that it can be explored exhaustively. That is, each AM can try all combinations of m_{ij}, v_{ij} and see which values do minimize $\mathrm{E}\left[T_{ij}\right]$.

F. Stability

A common pitfall in decentralized optimization mechanisms, such as the one presented in this paper, is the possibility for *oscillation* (see eg. [23], [24]). We address this concern by using the optimal answers provided by (11) to *guide* an

incremental bidding strategy, thus dampening oscillations and ensuring convergence. During each epoch, each AM calculates the optimal m_{ij} and v_{ij} based on the level of competition and resource availability in each EZ. Instead of using these directly, AMs use approximations $\hat{m}_{ij}$ and $\hat{v}_{ij}$ that are designed to *vary without risking large oscillations*, meaning, free from large step changes. We do this by limiting the change in $\hat{m}_{ij}$ and $\hat{v}_{ij}$ from one epoch to the next. Furthermore, we also ensure that both $\hat{m}_{ij}$ and $\hat{v}_{ij}$ remain bounded by specifying respective maximal values $\hat{m}_{\max}$ and $\hat{v}_{\max}$.

Three approaches are taken. The first approach is to move the number of bids at most one unit towards the optimal number. The second is to move the number of bids only a fraction of the way towards the new optimal and the third is to do this but reduce the fraction as time progresses.

Approach 1: This changes the number of bids by +1 (-1) if it is too low (high) with probability p, and leaves it unchanged with probability $1-p$. We used $p = .5$ in our evaluation.

Approach 2: This changes the number of bids in a manner based on the exponentially weighted moving average. The bid is increased (or decreased) by an amount proportional to the difference between the previous bid and the ideal bid. That is, the bid increases (or decreases) by $w(m_{ij} - m'_{ij})$ where m'_{ij} is the previous bid and $0 < w < 1$ is some constant. The nearer w is to 0 the more conservative the system is. Since bids must be whole numbers then the fractional part is treated as a probability, for example if $w(m_{ij} - m'_{ij}) = 1.2$ then the bid will increase by 1 (with probability p) or 2 (with probability $1-p$). We used $p = .8$ in our evaluation.

Approach 3: This is like Approach 2 but the factor w is decreased each round by multiplying it by some constant $0 < d < 1$. So, as epochs progress the changes to bids become more conservative.

It was found that the final results were similar in almost all cases investigated and the eventual position was insensitive to the convergence approach used (although on occasion the transient positions early in the simulation could be quite different). This is discussed further in Section IV-D. The results presented in the paper are for Approach 2 with $w = 0.1$.

The following section presents the simulation set up and the results of the evaluation of our proposed mechanism.

IV. Evaluation

We evaluate our mechanism through simulation. The simulation proceeds by epochs, with one epoch consisting of a *bidding phase* in which AMs decide, independently and in parallel, how many bids to submit to each EZ and at what valuation. These values are accumulated at each EZ, and when all required values are available, the system moves on to the *allocation phase*. At this point, EZs, independently and in parallel, perform resource allocation on the available bids and calculate the QoS experienced by the users of each service in their designated population. Statistics are then gathered and made available to AMs so that they can adapt their bidding behavior.

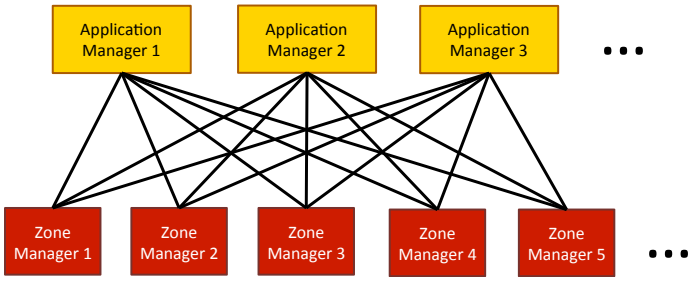


Fig. 3. Simulator architecture

A. Simulator Setup

Each application and ZM in our simulator is implemented as an independent thread and triggered by means of a thread pool [25]. As already stated, simulation progresses by discrete time steps denoted as *epochs*. Each epoch consists of two steps, and each one of these steps is implemented as a *map/reduce* operation. Simulation state is synchronized after each *reduce* step. We now broadly explain the two main steps of each simulation epoch by focusing in turn on the EZs and the AMs (see Fig. 3). The first step, denoted as the *execution step*, implements functionality to allow EZs to receive bids from all AMs, choose the winning set of bids, and calculate the QoS experienced by application users. Each EZ can then report to each AM the total number of bids N_i it received, the total number of service slots M_i it offers, and the distribution $F^{V^i}(v_{ij})$ of its received bid values. The second step, denoted as the *bid planning step*, implements functionality to allow each AM to evaluate (1), (6) and (7) as a function of m_{ij} and v_{ij}. Then, each AM can determine the optimal m_{ij}, v_{ij} pair that minimizes its expected total cost as expressed by (8), and use this information to drive its bidding behavior. A summary of the main simulation steps is provided below.

1) Initialize simulation - generate EZs and AMs according to QoE tolerance and population density distributions
2) Execution step
   a) AMs make bids and send to EZs
   b) Winning bids determined and results disseminated
3) Bid planning step
   a) AMs calculate the number and value of bids they would optimally make next round (if no other bids changed)
   b) AMs adjust their bidding towards this optimal number/value using Approach 1, 2 or 3 from previous section
4) Return to step 2)

In order to capture the different incentives behind service deployment decisions, we consider three different kinds of AMs, which we denote as *QoS-sensitive*, *Price-sensitive* and *Background*. Whereas the first two solve (11) to arrive to the optimal QoS/resource cost tradeoff embodied by (8), *Background* AMs generate bids and bid values drawn from static probability distributions. Hence, *Background* helps us model additional EZ load that is non-adaptive. Regarding *QoS-sensitive* and *Price-sensitive* AMs, the only difference is the

TABLE II
SIMULATION PARAMETERS

N_EZ	Number of execution zones	20
N_App_QoS	Number of *QoS-sensitive* apps ($\zeta = 30000$)	10, 20
N_App_Price	Number of *Price-sensitive* apps ($\zeta = 5000$)	10, 20
N_App_BG	Number of *Background* apps	10, 20
N_Srv_Slots	Number of service slots M_i in an execution zone	50, 100
Mu_Max Mu_Min	Maximum and minimum values for the rate at which a given execution zone can process user requests	5.5 3.5
Lambda_Max Lambda_Min	Maximum and minimum values for the rate at which user requests for a given application are generated at a given execution zone	1 1
N_Iterations	Maximum number of epochs for a given simulation replica	100
N_App_Threads	Number of application manager threads in the bid planning pool	450
N_EZ_Threads	Number of application manager threads in the execution pool	450
N_Replicas	Number of simulation replicas per simulation run	1
N_Hist_Bins	Number of histogram bins to represent $F^{V^i}(v_{ij})$	60

chosen value of ζ. For *QoS-sensitive* AMs we use $\zeta = 30000$; for *Price-sensitive* AMs we use $\zeta = 5000$. These values are arbitrary and only chosen to differentiate between these two categories of AMs.

We parametrize our simulations in terms of the variables presented in Table II. In order to reliably test the statistical properties of our simulation, we organize our simulations into *runs*, with each run consisting of a set of N_Replicas simulation replicas. For each one of these replicas we use the same values for all parameters and report the averages. For each simulation run, we choose a combination of individual values of N_EZ, N_App_QoS, N_App_Price, N_App_BG and N_Srv_Slots; the rest of the parameters are kept unchanged for all simulation runs. The values used for these variables are shown in the third column of Table II. In order to represent the probability densities $F^{V^i}(v_{ij})$, we used histograms that split the entire range for v_{ij} into N_Hist_Bins equally-sized bins. Additional information on our simulation setup can be found below.

B. Competition, QoS and Pricing

The first variable that we examine is the average cost incurred by each AM. Each subgraph in Fig. 4 corresponds to a combination of values as defined in Table II (see the top- and right-most edges of each panel). Because resource allocation is performed by means of a Vickrey auction, the total cost is defined by the losing bids in each EZ (the reader is reminded that in our multi-item Vickrey auction model a bidder winning n bids will pay the sum of the first n losing bids). Hence, in situations where there is a significant overprovisioning of resources, cost will remain close to zero. Such is the case for N_App_QoS = 10, N_App_Price = 10 and N_App_BG = 10; in this case, the cost paid is very low for all AM classes. As competition increases, however, we see that *Quality-sensitive* managers are much more aggressive in trying to secure EZ resources, and consequently end up paying higher costs. Although *Price-sensitive* AMs are more willing to tolerate higher blocking probabilities, they still implement a cost-quality tradeoff and hence experience consistently lower blocking probabilities than *Background* applications (see Fig. 5). It is interesting to note that the increase in total cost is not related only to competition with other adaptive AMs; even an increase in *Background* leads to an increase in total cost. This happens because adaptive applications will respond to increased competition to *Background* applications according to their cost-benefit tradeoff.

With regards to client blocking probability, *Quality-sensitive* managers achieve consistently better quality than *Price-sensitive* AMs (and, consequently, than *Background* services). It is interesting to note that, in the early stages of all simulation runs (lower values for the *epoch*), *Background* applications achieve lower blocking probabilities than either *Price-sensitive* or *Quality-sensitive* managers. This is due to the fact that these two categories of AMs progressively increase their number of bids for each EZ and their bid values. Hence, whereas at the beginning *Background* applications occupy a majority of the available service slots, they are soon displaced by adaptive AMs. This is shown explicitly in Fig 6. Although *Background* managers send an approximately constant number of bids to each EZ the number of bids they actually win decreases significantly as both *Price-sensitive* and *Quality-sensitive* adapt to the increased competition. However, this adaptation does not consist on indiscriminately increasing the number of sent bids; our approach allows managers to concentrate their resources on those EZs that will bring them a greater benefit (a greater decrease in client blocking probability) at a smaller expected resource cost. For example, in the case in which N_App_QoS = 10, N_App_Price = 10 and N_App_BG = 20, the number of bids sent by *Price-sensitive* and *Background* managers is very similar. However, the blocking probability experienced by *Price-sensitive* applications is much lower than that experienced by *Background* applications. There are two reasons for this. The first one is that, as shown in Fig. 6, *Price-sensitive* managers are winning more bids than *Background* managers; the second one, that our approach drives adaptive AMs to bid more in those EZs that provide better cost/benefit.

C. Scalability Considerations

One of the challenges faced by many auction-based systems is that they can exhibit high communications and/or computational complexity [17]. This means that, depending on the way in which the auction is resolved, a very large number of either messages or computations are required to achieve an optimal result. Hence, many of our engineering choices will be aimed to reducing the communications and computational overhead of the system. First, as detailed above, we eschew global optimality in which a central optimizer performs the optimal allocation in favour of a system based on local optima for each EZ. This breaking of the entire market into a number of *local markets* allows the computational problem to be massively parallelized, and has been shown to impose a tolerable impact on the quality of the solution [26]. Our use of service slots to

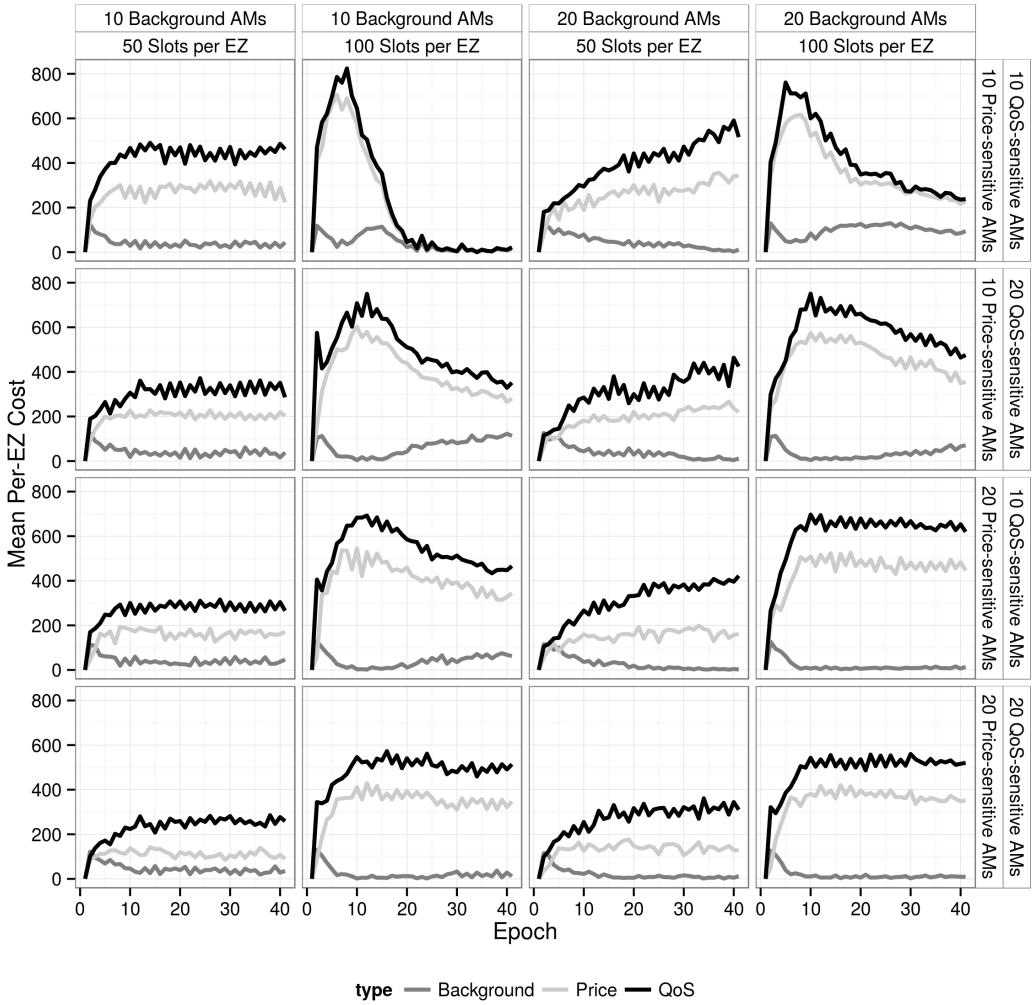


Fig. 4. Per-epoch application manager cost (arbitrary units)

model the way in which computational resources are offered to applications is also useful to scalability. In addition to discretizing the resource allocation problem, it also provides a simplified way in which the execution zones can advertise their available capacity and hence reduces the size of the messages required to that effect. Another benefit of the use of service slots is that it obviates the need for a bidding language to describe the various different bundles available in a multi-item auction; in our case, only the number of service slots required is needed. Finally, we constrain each bidder from assigning a bid value to each service slot they bid for; instead, bidders specify a single price and the number of service slot for which they bid at that price. This means that, when bidding for resources, each AM only needs to send the number of service slots bidded for and the value to be used for these bids. If the number of bids is constrained to an 8-bit integer and the value to bid is discretized to one of $\sim 2^8$ levels, a single 16-bit word is sufficient for each bid.

Another common problem with auction-based systems is that bidders can experience unpredictable results, depending on the number and value of bids received by the auctioneer. To help AMs reduce their uncertainty, we provide closed-form solutions of the expected outcomes of interest to AMs that are easy to calculate and depend on few inputs. In particular, AMs can evaluate their cost-benefit for each EZ only based on knowledge of the CDF of bid values for each auctioneer, the number of bids received by it, and the number of service slots it has on offer. By conveniently discretizing these to $\sim 2^8$ levels, the entire CDF can comfortably fit in a single IP packet. In particular, assume that each EZ reports 2^8 bin values and that these have been normalized so that the count per bin can be represented again with $\sim 2^8$ levels. Then, a full CDF can be represented in 2^{11} bits with no compression, which amounts to a $\sim$256 byte data structure. Since each AM will receive one such packet from each EZ, this amounts to a traffic stream of $\sim$2 $\times$`N_EZ` kilobits/epoch. For a representative value of `N_EZ`=1000 and and an epoch of 10 seconds, this amounts to 200 kbps.

With regards to computation, a conventional laptop dual-core CPU running at 2.50GHz achieves a throughput of $\sim$1500 simulation units per second, where each unit includes both bid planning for a single AM and Vickrey auction resolution for a single EZ. Even if we disregard CPU time devoted to auction resolution, this still implies a throughput of $\sim$1500 EZs per second using low-end hardware. Hence, a load of `N_EZ` = 1000 with an epoch length of 10 seconds as suggested

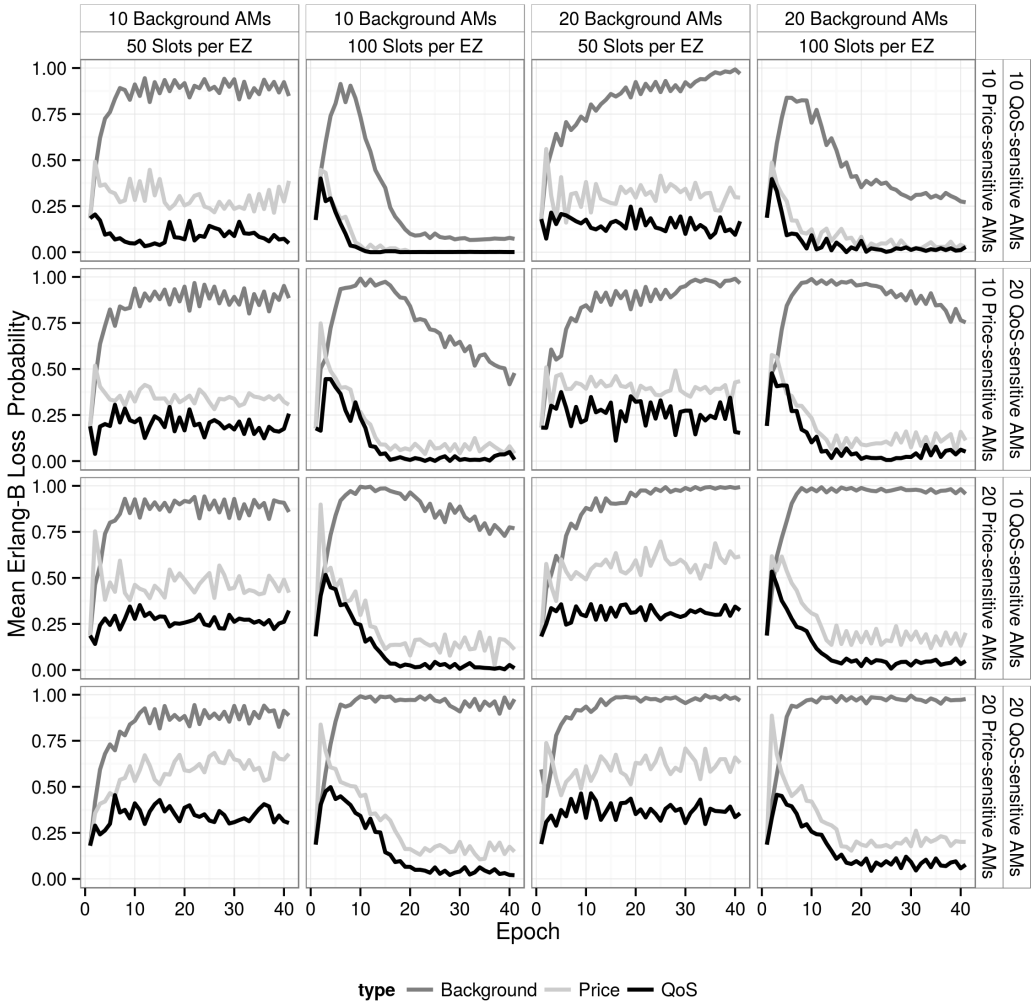


Fig. 5. Per-epoch execution zone blocking probability

above would induce a CPU load of only ∼7% on our reference machine. Based on these calculations, a virtual machine with 4 cores running at 2.50GHz could then act as AM for a system with N_EZ ≈ 60,000; we posit that this makes our system economical.

D. Discussion

The convergence approaches used (see Section III-F) were all tested in simulation with a variety of parameters. For reasons of space, only one of these can be presented in this paper. The cases studied were Approach 2 with values of w between 0.05 and 0.5 and Approach 3 with similar values for w but with d between 0.9 and 0.99. In these cases the final results (after all iterations) were nearly identical. Two exceptions were noted. Using Approach 3 (where the rate of change of bids shrank with time) the algorithm could simply stop moving bids too soon and end on a very different result if w shrank quickly (low values of d). This is as would be expected since that algorithm is designed to stop all changes in bidding as time progresses. With Approach 2 then higher values of w (for example 0.5) give end results which oscillate slightly more than the results shown but still giving a similar pattern overall. Higher values of w failed to converge but this is completely expected as that algorithm allows extremely severe changes every iteration.

In all cases studied the transient results for the first ten to twenty epochs differed with the parameters. The major difference is observed in the number of sent bids. The results in this paper are all for Approach 2 with $w = 0.1$, which show an initial "overshoot". If Approach 1 is used instead, this overshoot is substituted for a slow climb; this happens because Approach 1 seeks the equilibrium position more slowly. This is best illustrated by the results for 10 background slots per AM, 100 slots per EZ (the second column in all results). In these figures a transient can be seen in epochs 5–20 which is not present in slower converging methods for reaching equilibrium. If $w = 0.5$ is used with Approach 2, this overshoot is much larger. In most cases, however, the difference between the three approaches was for the first 10 or 20 epochs only.

Our approach assumes that bidding cycles, comprised of multiple epochs, are undertaken in the context of stable traffic demand such that bidders remain a fixed group, and their internal service slot valuations remain fixed. We refer to [27] where dynamically changing traffic patterns can be considered constant within 10 minute intervals. Our results show that

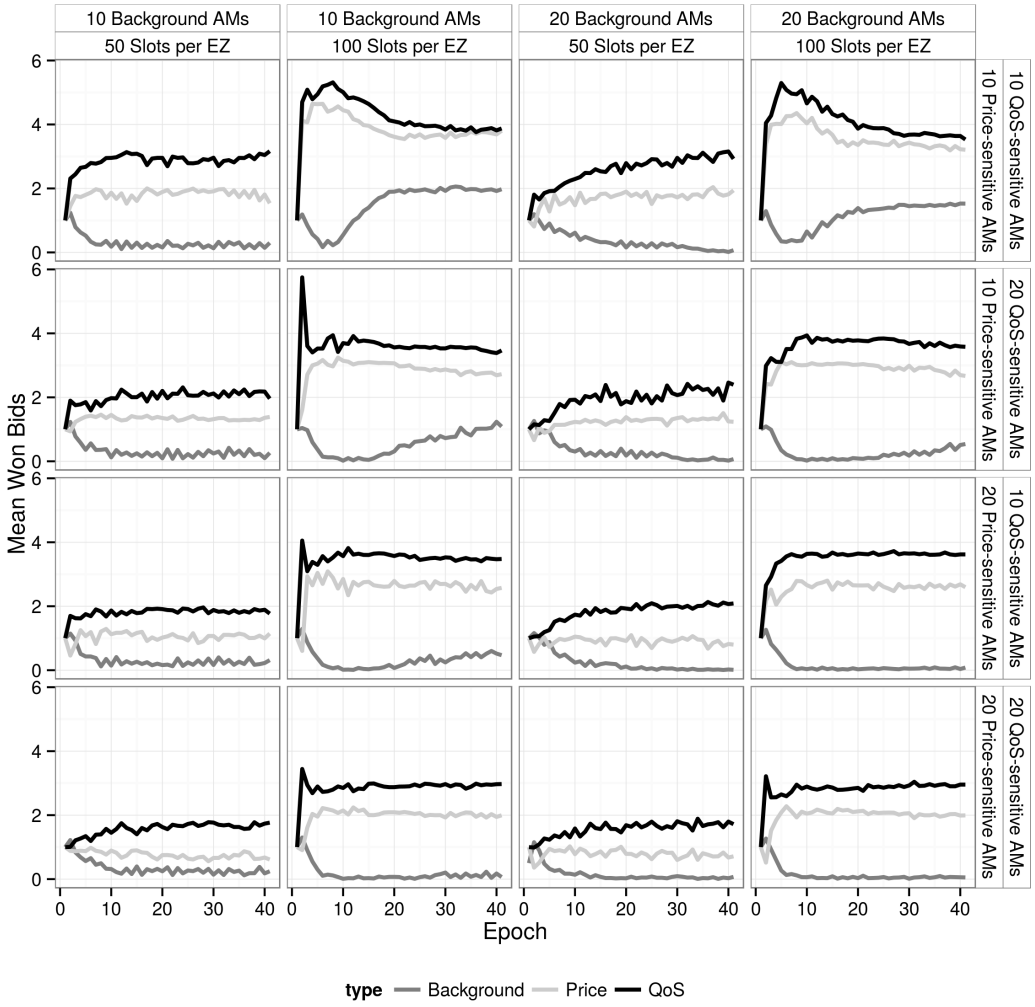


Fig. 6. Per-epoch average won bids for application managers of given type (thousands)

the bidding process converges at ~ 20 epochs, which is well within the period of traffic stability with epochs of 10s as we have assumed in this paper.

We have made every effort to keep the number of parameters used in the model to a minimum in order to reduce the parameter space. However, it remains a model with a number of free parameters. Of these, our plots explore the most important: the number of applications in the ecosystem (in terms of background, price sensitive and QoS sensitive). The number of iterations per simulation and the number of simulation replicas are only important to ensure convergence is reasonably likely and that the results of a run are not merely a result of random processes; they do not provide interesting information per-se. This is also the case for other parameters such as μ, λ their ratio ρ; the number of EZs; and the ratio of ζ values between different types of adaptive applications. All of these are fixed for our simulations in the interest of space because they did not provide additional information.

As would be expected the results showed that AMs focussing on QoS ended up paying more than those focusing on reducing price. This difference was greater when more AMs were competing for fewer resources. In return those AMs prioritising QoS obtained a lower loss probability. In extreme cases, the background AMs which did not vary their behavior obtained extremely poor loss probabilities (nearing 100%) when there was a great deal of competition for a small number of slots. This was because they won very few bids (indeed in some scenarios their mean number of won bids was far less than 1 – that is in a typical epoch they won no bids). Only in the scenarios where there was very little competition did the QoS and the price sensitive AMs obtain similar results to each other. In particular when there are 10 AMs of each type competing for 100 EZ slots then there is little overall loss and the QoS and price sensitive AMs end up with very similar costs and QoS.

V. Related Work

Due to the useful properties that we have briefly described elsewhere in the paper, there is a large body of work on auction-based resource allocation. General analysis of the topic can be found in works focusing on the game theoretic aspects of multi agent systems, such as [28] sec. 11.2.3 and [17] sec. 9.3 to name but two examples. We now provide concrete and representative examples of related work in the literature, and how they relate to our current proposal.

In [29] the authors describe a double auction scenario between mobile users and cloudlets. Examples of these kind

rely on a trusted third party to administer the trading. Our approach is architecturally different in that a logically centralized matchmaking entity is not involved in the resource allocation decision; instead, the system is self-organized. There have also been proposals for decentralized markets. An example of this is CompuP2P [6], a system that implements an open market for peer resources quantized into different markets. Each market is managed by a particular peer through a dynamic hash table (DHT), and pricing is arrived at by using Vickrey auctions [16]. Our proposed architecture relies on direct communications rather than on a DHT, thus simplifying the protocols and avoiding the need for costly and time-consuming overlay maintenance operations. Vickrey auctions have also been used for virtual network embedding [30], where service providers lease resources from multiple infrastructure providers for constructing virtual topologies across autonomous systems. In this case, infrastructure providers bid for the virtual resources they are willing to host according to the requirements of the service provider. A distinct difference with our work is that bidding concerns communication resources as opposed to compute resources, but also the fact that the physical resource providers are the bidders rather than the auctioneers. Another system based on Vickrey auctions is Spawn [7], which uses an open market to solve a distributed CPU resource allocation problem. Rather than true money, Spawn uses an abstract form of priority, so that better funded processes can obtain correspondingly better access to the computing infrastructure than others. A limitation of Spawn is that it may not scale well with spatial price dynamics; the scalability of our approach under price variability among EZs is guaranteed by the auction resolution process and demonstrated in simulation.

Some contributions to the literature have moved away from simple Vickrey auctions to consider multi-item auctions. Full-blown combinatorial auctions, in which bidders express their preferences for product bundles, are more versatile but can be very computationally intensive and involve large delays [17]. Hence, multiple simplifications have been proposed. In [10], each participant allocates its finite budget to bid on a given resource set, and receives a proportion of each resource commensurate with the proportion of its bid with the bids of other participants; this same technique was later on proposed by [11] as a replacement for the unchoking policy of BitTorrent. Another auction-based resource management implementation is Mirage [12], where combinatorial auctions using a centralized virtual currency environment are used for sensor network testbed resource allocation. Our work presents yet another take on computationally tractable multiunit auctions which is efficient, as evidenced by our simulation results.

A first set of server placement techniques has been developed in the context of data centers. Their key rationale is to minimize data center energy consumption by maximizing the utilization of the running servers through efficient placement of the work load (in the form of virtual machines). Various decision parameters have been studied, such as the workload of the virtual machines or their memory consumption [31] [32] or the type of resources requested [13]. However, these algorithms are not directly applicable to our problem definition, as in the data center case the effect of the underlying physical network topology is negligible: inside data centers, dedicated high-bandwidth and low-latency links can be used. Another contribution in this direction is [14], where the authors propose a centralized service placement algorithm requiring information on the entire network topology. The algorithm maximizes the satisfied demand and minimizes the number of placement changes. In [33] and [34], the authors propose a decentralized variant of this algorithm that overcomes the scalability limitations of [14] by performing a local optimization on each server, based on state information from that servers neighborhood. A decentralized solution was also proposed in [35], which uses a round-based gossip protocol for exchanging state information between servers. In an effort to combine the optimality and scalability benefits of centralized and distributed solutions, respectively, a hierarchical model was investigated in [36] and [37]. However, the above approaches do not take into account latency or any other characteristics of the underlying physical network. A dynamic and latency-aware service placement algorithm for assigning resources to services is presented in [38]. The servers are connected via a peer-to-peer overlay technology. Each server is responsible for both taking part in the management tasks and running a subset of the available services. A comprehensive review of various service placement approaches can be found in [39]. In general, resource allocation in non-auction based approaches is predominantly driven by the objectives of the infrastructure provider, eg. balancing resource usage, reducing energy consumption, etc. The absence of customer influence in this process can compromise the expected service quality but also allow providers to dictate the price of resources, which might not be fair.

The techniques discussed above require a mapping of individual workloads (services) to a set of servers. Within the field of distributed systems, the replica placement problem has also been studied extensively, i.e. where to cache or replicate popular data objects [15]. A model taking QoS into consideration is presented in [40]. Parameters like storage cost, update cost and access cost of data replication are used for the replica placement. Being NP-complete, two heuristic algorithms are presented for the model. For tree networks, two new policies for QoS-aware placement are presented in [41].

VI. Conclusions

Cloud applications have different requirements in terms of cost and quality, and different applications will coexist in the same cloud infrastructure that have different tradeoffs between these. The mechanism presented in this article allows different applications to implement their own tradeoffs transparently by abstracting away the task of allocating resources between applications with differing requirements to an auction mechanism. This allows the system to be responsive to the needs of each service, without the need for specialized treatment. In addition, because applications themselves use the prices provided by the system as signals to drive their own cost-quality tradeoffs, they experience better outcomes than those that would be possible in a static system.

Many properties of the system are predictable, because they are rooted on closed-form solutions of the Vickrey auction, a

well-researched mechanism. In addition, we carry out extensive simulations to verify that the system as a whole (that is, the global market in which applications bid for resources according to their needs) is both stable and efficient. We find that it is stable in the sense that, if the input parameters of the system do not change, it converges to an equilibrium state. It is efficient in the sense that resources are allocated to those applications that need them the most, and in those EZs where they provide the best improvement to the quality experienced by its users.

In the process of reaching an efficient equilibrium, the system will track any changes in its inputs, and hence, it will adapt to the changing needs of those applications using it. This provides self-tuning, that is, the ability of the system to naturally configure its resources to best fit the demand profiles imposed by heterogeneous populations of cloud users.

ACKNOWLEDGMENT

This research has received funding from the Seventh Framework Programme (FP7/2007-2013) of the European Union, through the FUSION project and the FLAMINGO Network of Excellence (grant agreements 318205 and 318488).

REFERENCES

[1] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *SIGCOMM Comput. Commun. Rev.*, vol. 44, no. 5, pp. 27–32, Oct. 2014. [Online]. Available: http://doi.acm.org/10.1145/2677046.2677052

[2] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*. New York, NY, USA: Cambridge University Press, 2007.

[3] K. Krauter, R. Buyya, and M. Maheswaran, "A taxonomy and survey of grid resource management systems for distributed computing," *Softw. Pract. Exper.*, vol. 32, no. 2, pp. 135–164, Feb. 2002.

[4] W. Wang and B. Li, "Market-driven bandwidth allocation in selfish overlay networks," in *Proc. of IEEE INFOCOM*, 2005, pp. 2578–2589.

[5] R. Landa, R. Clegg, E. Mykoniati, D. Griffin, and M. Rio, "A Sybilproof Indirect Reciprocity Mechanism for Peer-to-Peer Networks," in *Proc. of INFOCOM*, 2009.

[6] R. Gupta, V. Sekhri, and A. K. Somani, "CompuP2P: An Architecture for Internet Computing Using Peer-to-Peer Networks," *IEEE Trans. Parallel and Distrib. Syst. (TPDS)*, vol. 17, no. 11, 2006.

[7] C. A. Waldspurger, T. Hogg, B. A. Huberman, J. O. Kephart, and W. S. Stornetta, "Spawn: A Distributed Computational Economy." *IEEE Trans. Software Eng.*, vol. 18, no. 2, pp. 103–117, 1992.

[8] J. Shneidman, C. Ng, D. C. Parkes, A. AuYoung, A. C. Snoeren, A. Vahdat, and B. Chun, "Why Markets Could (But Don't Currently) Solve Resource Allocation Problems in Systems," in *Proc. of ACM HotOS*. USA: USENIX - (TCOS), June 2005.

[9] Y. Fu, J. Chase, B. Chun, S. Schwab, and A. Vahdat, "SHARP: an architecture for secure resource peering," in *Proc. of ACM Symp. on Op. Sys. Princ. (SOSP)*, 2003.

[10] M. Feldman, K. Lai, and L. Zhang, "A Price-Anticipating Resource Allocation Mechanism for Distributed Shared Clusters," in *Proc. ACM EC*, Jun. 2005.

[11] D. Levin, K. LaCurts, N. Spring, and B. Bhattacharjee, "BitTorrent is an auction: analyzing and improving BitTorrent's incentives," in *Proc. of SIGCOMM*, 2008, pp. 243–254.

[12] B. N. Chun, P. Buonadonna, A. AuYoung, C. Ng, D. C. Parkes, J. Shneidman, A. C. Snoeren, and A. Vahdat, "Mirage: A Microeconomic Resource Allocation System for SensorNet Testbeds," in *Proc. of EmNetsII*, 2005.

[13] M. Stillwell, D. Schanzenbach, F. Vivien, and H. Casanova, "Resource allocation algorithms for virtualized service hosting platforms," *J. Parallel Distrib. Comput.*, vol. 70, no. 9, pp. 962–974, Sep. 2010.

[14] A. Karve, T. Kimbrel, G. Pacifici, M. Spreitzer, M. Steinder, M. Sviridenko, and A. Tantawi, "Dynamic placement for clustered web applications," in *Proc. of ACM WWW*, USA, 2006, pp. 595–604.

[15] S. Zaman and D. Grosu, "A Distributed Algorithm for the Replica Placement Problem," *IEEE Trans. Parallel and Distrib. Syst. (TPDS)*, vol. 22, no. 9, pp. 1455–1468, 2011.

[16] W. Vickrey, "Counterspeculation, Auctions, and Competitive Sealed Tenders," *The Journal of Finance*, vol. 16, no. 1, pp. 8–37, 1961.

[17] P. Cramton, Y. Shoham, and R. Steinberg, *Combinatorial Auctions*. MIT Press, 2006.

[18] M. Yokoo, "False-name bids in combinatorial auctions," *ACM SIGecom Exchanges*, vol. 7, no. 1, pp. 48–51, 2007.

[19] Q. Wang, B. Ye, B. Tang, S. Guo, and S. Lu, "ebay in the clouds: False-name-proof auctions for cloud resource allocation," in *Distributed Computing Systems (ICDCS), 2015 IEEE 35th International Conference on*. IEEE, 2015, pp. 153–162.

[20] H. A. David and H. N. Nagaraja, *Order Statistics*, ser. Wiley Series in Probability and Statistics. Wiley-Interscience, 2003.

[21] N. Balakrishnan and C. R. Rao, *Order Statistics: Theory and Methods*, ser. Handbook of Statistics. Elsevier, 1998, vol. 16.

[22] D. Bertsekas and R. Gallager, *Data Networks*, 2nd ed. P. Hall, 1992.

[23] D. Bertsekas, "Dynamic behavior of shortest path routing algorithms for communication networks," *IEEE Trans. on Auto. Control*, vol. 27, no. 1, pp. 60–74, 1982.

[24] Z. Wang and J. Crowcroft, "Analysis of Shortest-path Routing Algorithms in a Dynamic Network Environment," *SIGCOMM Comput. Commun. Rev.*, vol. 22, no. 2, pp. 63–71, Apr. 1992.

[25] R. P. Garg and I. Sharapov, *Techniques for Optimizing Applications - High Performance Computing*. Prentice-Hall, 2002.

[26] S. M. Kakade, M. Kearns, and L. E. Ortiz, "Graphical Economics," *Learning Theory (Springer LNCS)*, vol. 3120, pp. 17–32, 2004.

[27] P. Wendell, J. W. Jiang, M. J. Freedman, and J. Rexford., "DONAR: Decentralized Server Selection for Cloud Services," in *SIGCOMM*, 2010.

[28] Y. Shoham and K. Leyton-Brown, *Multiagent Systems: Algorithmic, Game-Theoretic, and Logical Foundations*. Cambridge University Press, 2009.

[29] A. Jin, W. Song, P. Wang, D. Niyato, and P. Ju, "Auction Mechanisms Toward Efficient Resource Sharing for Cloudlets in Mobile Cloud Computing," *Transactions on Services Computing*, 2015.

[30] F.-E. Zaheer, J. Xiao, and R. Boutaba, "Multi-provider service negotiation and contracting in network virtualization," in *Network Operations and Management Symposium (NOMS), 2010 IEEE*, 2010, pp. 471–478.

[31] J.-W. Jang, M. Jeon, H.-S. Kim, H. Jo, J.-S. Kim, and S. Maeng, "Energy Reduction in Consolidated Servers through Memory-Aware Virtual Machine Scheduling," *IEEE Trans. on Computers*, vol. 60, no. 4, pp. 552–564, 2011.

[32] M. B. Reynolds, D. R. Hulce, K. M. Hopkinson, M. E. Oxley, and B. E. Mullins, "A Bin Packing Heuristic for On-Line Service Placement and Performance Control," *IEEE Transactions on Network and Service Management*, vol. 10, no. 3, pp. 326–339, 2013.

[33] C. Adam, R. Stadler, C. Tang, M. Steinder, and M. Spreitzer, "A Service Middleware that Scales in System Size and Applications," in *Proc. of IFIP/IEEE IM (Intl. Symp. on Int. Net. Man.)*, 2007, pp. 70–79.

[34] C. Adam and R. Stadler, "Service middleware for self-managing large-scale systems," *IEEE Transactions on Network and Service Management*, vol. 4, no. 3, pp. 50–64, 2007.

[35] F. Wuhib, R. Stadler, and M. Spreitzer, "A gossip protocol for dynamic resource management in large cloud environments," *IEEE Transactions on Network and Service Management*, vol. 9, no. 2, pp. 213–225, 2012.

[36] H. Moens, J. Famaey, S. Latré, B. Dhoedt, and F. De Turck, "Design and evaluation of a hierarchical application placement algorithm in large scale clouds," in *Proc. of IFIP/IEEE IM (Intl. Symp. on Int. Net. Man.)*, 2011, pp. 137–144.

[37] H. Moens and F. De Turck, "A scalable approach for structuring large-scale hierarchical cloud management systems," in *Proc. of IEEE/IFIP CNSM (Int. Conf. on Net. and Serv. Man.)*, 2013, pp. 1–8.

[38] J. Famaey, W. De Cock, T. Wauters, F. De Turck, B. Dhoedt, and P. Demeester, "A latency-aware algorithm for dynamic service placement in large-scale overlays," in *Proc. of IFIP/IEEE IM (Intl. Symp. on Int. Net. Man.)*, 2009, pp. 414–421.

[39] B. Jennings and R. Stadler, "Resource management in clouds: Survey and research challenges," *Journal of Network and Systems Management*, pp. 1–53, 2014.

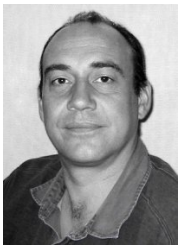
[40] C.-W. Cheng, J.-J. Wu, and P. Liu, "QoS-aware, access-efficient, and storage-efficient replica placement in grid environments," *J. Supercomput.*, vol. 49, no. 1, pp. 42–63, Jul. 2009.

[41] A. Benoit, V. Rehn-Sonigo, and Y. Robert, "Replica Placement and Access Policies in Tree Networks," *IEEE Trans. Parallel and Distrib. Syst. (TPDS)*, vol. 19, no. 12, pp. 1614–1627, 2008.

Raul Landa is currently a Data Scientist in the Network Planning and Operations group within Sky UK Network Services. His research interests involve the measurement-based modelling of Internet video quality of experience, the large-scale behaviour of content delivery networks and the strategic aspects in peer-to-peer overlays. He received a PhD in Electronic and Electrical Engineering from University College London, UK.

Marinos Charalambides is a senior researcher at University College London. He holds a BEng, a MSc and a Ph.D. from the University of Surrey, UK. He has been working in a number of European and UK national projects since 2005 and his research interests include network programmability, content distribution, and adaptive resource management. He has been the technical program chair of several events and serves in the committees of the main network and service management conferences.

Richard G. Clegg is a Research Fellow at the Department of Computing in Imperial College London. His PhD in mathematics and statistics from the University of York was gained in 2005. His research interests include investigations of the dynamic behaviour of networks and measurement of network traffic statistics.



David Griffin is a Principal Research Associate at University College London. He has a PhD in Electronic and Electrical Engineering from UCL. His research interests are in planning, management and dynamic control for providing QoS in multi-service networks, p2p networking and novel routing paradigms in the Internet.

Miguel Rio is a Reader (Associate Professor) in Computer Networks at the Department of Electronic and Electrical Engineering at University College London. He has published extensively in top ranked Conferences and journals in the areas of Network measurement, congestion control, new network architectures and, more recently, in the interaction between cloud and network services. He holds a PhD from the University of Kent at Canterbury and MSc and MEng from the Department of Informatics, University of Minho, Portugal.