A Dynamic Resource Overbooking Mechanism in Fog Computing

Fuming Zhang*, Zhiqing Tang*, Mingcheng Chen*, Xiaojie Zhou*, Weijia Jia†*
*Department of Computer Science and Engineering, Shanghai Jiao Tong University, Shanghai 200240, China
†Centre of Data Science, University of Macau, SAR Macau 999078, China
{zhangfuming-alex,domain,35052519971211001x,szxjzhou}@sjtu.edu.cn, jia-wj@cs.sjtu.edu.cn

***Abstract*—Fog Computing (FC - similarly edge computing) as new computing paradigm can support distributed domain-specific or area-specific applications with cloud-like quality of service (QoS). This promising paradigm thus can find its wide applications in various industrial scenarios and smart cities in which the resource requirements will be divided into peak-hour or non-peak-hour. To deal with such features of applications, a flexible resource allocation approach based on pricing model can be critical for the success of such paradigm. To the best of our knowledge, we have not seen such pricing based resource allocation approach ever been reported for FC scenarios. In this paper, we propose a novel pricing based dynamic resource allocation model through overbooking mechanism, and it is realized through three steps: 1) According to different QoS requirements of user tasks, methods of on-demand billing, daily billing, and auction billing are designed, in which we allow the resource to be overbooked; 2) For auction billing, we design an auction approach including pricing rule and winner determination rule. We prove that our auction approach guarantees individual rationality, computational efficiency, and truthfulness. 3) To overbook as much resource as possible with a high degree of QoS satisfaction of on-demand and daily billing, we overbook the resource based on a resource utilization prediction using neural network and service level agreement violation feedback. In the end, we validate the mechanism with real-world data trace. Experimental results show that our auction approach achieves desirable properties, and our dynamic resource overbooking mechanism maximizes the profit of nodes with a high degree of QoS satisfaction of on-demand and daily billing and a high resource utilization prediction accuracy rate.**

I. Introduction

In recent years, due to the rapid growth in the number of mobile devices, cloud computing, which is far apart from these devices, can not meet the needs of delay-sensitive applications like video streaming and face recognition. In fog computing (FC), computation resource of cloud data centers is partially offloaded to the decentralized fog (edge) nodes by deploying the nodes at the edge of the network [1]. Compared with cloud computing, decentralized nodes can not only support the mobility of tasks [2], but also significantly reduce delay while meeting the resource requirements of mobile tasks [3]. Besides, fulfilling delay requirements, FC can well support domain-specific large-scale distributed decision-making systems [4], like the smart transportation system in cities [5].

However, the delay requirements of resource-consuming applications may be different in FC. For example, in the smart transportation system, real-time traffic information processing tasks, which have apparent peak-hour and non-peak-hour, require long-term, low-overhead, and low-delay data processing. Sudden traffic accident information processing tasks also need to be processed in time but require less computation time. Both of the above delay-sensitive tasks are ideal scenarios for applying FC [5]. Also, the tasks of processing road surveillance video, which are not delay-sensitive but need much more computation resource, are suitable for execution on the nodes [6]. As a result, a flexible pricing model of the nodes is necessary for reconciling the mismatch between resource demands and resource capacity.

To the best of our knowledge, there are relatively few researches on the pricing model of FC. Bittencourt et al. [7] describe the pricing model of FC and propose a general architecture. They focus on discussing its components, interfaces and interactions but do not give a practical algorithm. Zhang et al. [8] propose a hierarchical Stackelberg game based pricing strategy to achieve high utilities with a 3-layer model in FC. However, the task request submitted by each user is the same. To fill in these gaps, we propose a pricing model consisting of on-demand, daily and auction billing methods. On-demand billing is suitable for unexpected delay-sensitive, short-time tasks. Daily billing is for long-term tasks with strict delay requirements. Moreover, auction billing is designed for computationally intensive and delay insensitive tasks. The nodes can rent out their resource for revenue, and incoming tasks can choose the billing method flexibly.

Among these three billing methods, auction [9] is a trendy way to deal with the relationship between different buyers and sellers effectively. The auction theory has been extensively studied. Jin et al. [10], [11] design two auction approaches in Cloudlets. However, both of them are based on homogeneous tasks. Wang et al. [12] consider heterogeneous tasks, but they aim to minimize the expenditure of users without considering the profit. Compared with the existing work, the auction approach proposed mainly has the following differences: 1) The heterogeneity of tasks (buyers) and servers (sellers) is considered. The servers of one node are grouped to receive the bids from tasks. 2) Each server can accept multiple (winning) tasks, and the remaining available resource which can be sold by auction is calculated dynamically. 3) Multiple resource requirements of tasks are taken into consideration, and our results are obtained based on these requirements.

Furthermore, we overbook resource to solve the problem

2155-6814/18/$31.00 ©2018 IEEE
DOI 10.1109/MASS.2018.00023

of resource waste during non-peak-hour. Specifically, among these three billing methods, on-demand and daily billing methods have a relatively high price and a high degree of quality of service (QoS) satisfaction. Unused resource will be sold again by auction at a relatively low price with a low degree of QoS satisfaction. Researches on resource overbooking to make better utilization of resource are common in cloud computing [13], [14]. However, the relevant research is still in its infancy in FC. Barbarossa et al. [15] propose a strategy to overbook the computation and communication resource based on the statistics of blocking events in mmW-mobile edge computing. Slim et al. [16] propose a costless service offloading strategy for distributed edge cloud considering resource overbooking.

To effectively overbook resource, the capacity of available resource needs to be dynamically determined. Moreno et al. [13] and Imam et al. [17] predict the resource utilization through neural network and overbook resource based on predicted results. However, they overbook resource with no regard for service level agreement violation (SLAV). To solve this, we use a long short-term memory (LSTM) [18] based neural network for prediction, and we propose a dynamic resource adjustment mechanism, which is similar to TCP congestion control [19]. It dynamically adjusts available resource and prices through SLAV feedback. Different from previous work, we overbook resource as much possible with a high degree of QoS satisfaction of on-demand and daily billing methods.

To summarize, this paper proposes a dynamic resource overbooking mechanism with a pricing model. This article mainly has the following contributions:

1) We design a novel auction approach by applying pricing and winner determination rules. We prove that our approach guarantees individual rationality, computational efficiency, and truthfulness.
2) We adopt novel resource overbooking and prediction algorithms based on on-demand and daily billing through LSTM based neural network and SLAV feedbacks.
3) We validate our algorithms with real-world data trace, and the experimental results show that our auction approach achieves desirable properties. And our dynamic resource overbooking mechanism can maximize the profit of nodes and achieve a high degree of QoS satisfaction of on-demand and daily billing with a high accuracy rate of resource utilization prediction.

The remainder of this paper is organized as follows. In Section II, we describe the FC architecture model and pricing model, then formulate the problem. The online auction approach is proposed and analyzed in Section III. Then, the dynamic resource overbooking mechanism is illustrated in Section IV. The experimental settings and results are described in Section V. Finally we conclude the paper in Section VI.

II. Modeling and Problem Formulation

In this section, we illustrate the system model for FC in II-A, which includes mobile tasks and fog nodes. Then we introduce the pricing model with on-demand, daily and auction billing methods in II-B. Finally, we formulate the problem in II-C.

A. System Model

We consider a general FC architecture. For the tasks of mobile users, fog service providers gain revenue by leveraging computation resource of nodes. Other agencies can rent these resource if they have pending tasks. Take the smart transportation system as an example, re-planning of traffic routes is a delay-sensitive task which requires long-term computation resource. Unexpected traffic accident information processing is also delay-sensitive but only requires short-term computation resource. Moreover, road surveillance video processing is not delay-sensitive but needs much more computation resource.

We assume that there is a set of mobile tasks $B = \{b_1, b_2, ..., b_n\}$ and a set of heterogeneous and distributed nodes $F = \{F_1, F_2, .., F_m\}$. Mobile users generate tasks and offload them to the nodes. The tasks are divided into delay-sensitive and computation-oriented tasks. The former needs to be processed in time, while the latter requires a lot of computation resource. More specifically, for the i-th task, $b_i = \{t_i^s, t_i^e, r_i, v_i\}$, where t_i^s is the estimated start time, t_i^e is the estimated end time, r_i is the resource request, and v_i is the valuation. For each $r_i = \{r_i.c, r_i.m\}$, $r_i.c$ and $r_i.m$ are the requests of CPU and memory resource, respectively. v_i contains the valuation for CPU and memory resource for each node, which are denoted as $\{v_i.c_1, v_i.c_2, ...v_i.c_m\}$ and $\{v_i.m_1, v_i.m_2, ...v_i.m_m\}$, respectively. Here, the valuation of each task is slightly different due to the difference in the distance from the fog node [20]. The submit time of b_i is t_i. For each F_i, there is a set of physical servers $S_i = \{S_i^1, S_i^2, ..., S_i^j\}$, where S_i^j is the j-th server of F_i. For each server S_i^j, we use $S_i^j.a_0.c$ and $S_i^j.a_0.m$ to denote CPU and memory resource capacity of S_i^j, respectively.

B. Pricing Model

The nodes could earn the profit by renting out computation resource, and users can choose different billing methods for different types of tasks. Traditionally, the nodes can only rent out as much resource as the servers own. But, we notice that the actual resource utilization is always much less than the resource allocated [21], [22]. To maximize the profit of the nodes, we put the resource that has been allocated but unused for a second sale, called resource overbooking [13].

For different types of tasks, we design three different billing methods as follows.

- **On-demand**: for tasks that are delay-sensitive with short duration. Such billing method has a high degree of QoS satisfaction with a considerable price.
- **Daily**: for tasks that are also delay-sensitive but with a long duration. In such billing method, computation resource is reserved, and bills are paid per day. Such billing method has a high degree of QoS satisfaction but with a relatively low price.
- **Auction**: for tasks that are insensitive to delay but need much more computation resource. Such billing method can provide computation resource with much lower price with a low degree of QoS satisfaction. These tasks have

limited running time and would be evicted when the resource is depleted.

Users can flexibly choose the billing method. For example, most of the resource allocated to traffic information processing tasks is unused during non-peak-hour, and can be used for a second sale through the auction. The degree of QoS satisfaction of each server $L(t) = \{L_i^j(t)|S_i^j \in F\}$ is defined as:

$$L_i^j(t) = \min\left\{\frac{S_i^j.a_0.c - S_i^j.o(t).c}{S_i^j.a_0.c - S_i^j.a(t).c}, \frac{S_i^j.a_0.m - S_i^j.o(t).m}{S_i^j.a_0.m - S_i^j.a(t).m}, 1\right\}, \tag{1}$$

where $S_i^j.a_0 = \{S_i^j.a_0.c, S_i^j.a_0.m\}$ is the resource capacity of S_i^j, $S_i^j.a(t) = \{S_i^j.a(t).c, S_i^j.a(t).m\}$ is the total unused resource of S_i^j at t, and $S_i^j.o(t) = \{S_i^j.o(t).c, S_i^j.o(t).m\}$ is the total overbooked resource. From Eq. (1) we can see that if there is no SLAV, the degree of QoS satisfaction equals to 1. Otherwise, it is less than 1.

C. Problem Formulation

Our goal is to maximize the profit of the nodes through overbooking with a high degree of QoS satisfaction for those on-demand and daily tasks, which is defined as the revenue minus the cost.

The revenue of nodes is the sum of the payments of on-demand, daily and auction tasks. For task b_i, if b_i is an on-demand task, its payment depends on its CPU and memory demands, which is defined as:

$$w_i^o = (r_i.c \times w_c^o + r_i.m \times w_m^o) \times |t_i^e - t_i^s|,$$

where w_c^o and w_m^o are the corresponding on-demand prices of the CPU and memory resource, respectively. t_i^s is the start time of b_i and t_i^e is the end time.

In addition, if b_i is daily task, the payment is defined as:

$$w_i^p = (r_i.c \times w_c^p + r_i.m \times w_m^p) \times t_d,$$

where w_c^p and w_m^p are the prices of daily methods of the CPU and memory resource, respectively, and t_d is the amount of days. In general, the prices of on-demand and daily tasks are fixed as constants [23].

Otherwise, if b_i is auction task, the payment is defined as:

$$w_i^a = \int_{t_i^s}^{t_i^e} (r_i.c \times p_i.c(t) + r_i.m \times p_i.m(t))\, dt,$$

where $p_i.c(t)$ and $p_i.m(t)$ are the payments of CPU and memory resource respectively. Their values are determined through the auction approach described in Section III.

Therefore, the total profit of S_i^j is defined as:

$$w_{total}^{i,j}(t) = d_L^{i,j}(t) \times \left(\sum_{b_k \in B_o^{i,j}} w_k^o + \sum_{b_k \in B_p^{i,j}} w_k^r\right) + \sum_{b_k \in B_a^{i,j}} w_k^a, \tag{2}$$

where $B_o^{i,j}$, $B_p^{i,j}$ and $B_a^{i,j}$ are the sets of on-demand, daily and auction tasks that run in server S_i^j, respectively. $d_L^{i,j}$ is a discount rate. To ensure the high degree of QoS satisfaction of on-demand and daily billing, we have to reduce the revenue to punish such SLAV with such a discount, which is defined as the SLA of Microsoft Azure [24]:

$$d_L^{i,j}(t) = \begin{cases} 1, & L_i^j(t) \geq 99.95\% \\ 0.9, & 99\% \leq L_i^j(t) < 99.95\% \\ 0.75, & 95\% \leq L_i^j(t) < 99\% \\ 0, & L_i^j(t) < 95\% \end{cases}.$$

Besides, the cost of the servers mainly consists of the energy consumption of CPU and memory utilization [25], which is defined as:

$$\begin{aligned} C_i^j(t) =& e \times [(S_i^j.a_0.c - S_i^j.a(t).c) \times h_c \\ &+ (S_i^j.a_0.m - S_i^j.a(t).m) \times h_m], \end{aligned}$$

where e is the unit price of electricity, h_c and h_m are the amounts of power consumed per unit of CPU and memory.

Thus the problem is formulated as:

Problem 1.

$$\max R = \sum_{S_i^j \in F} \left(\sum_{t=0}^{T} w_{total}^{i,j}(t) - C_i^j(t)\right), \tag{3}$$

$$\begin{aligned} s.t. \quad & S_i^j.o(t).c \leq S_i^j.a(t).c \quad \{i,j\} \in \{i,j|S_i^j \in F\} \\ & S_i^j.o(t).m \leq S_i^j.a(t).m \quad \{i,j\} \in \{i,j|S_i^j \in F\} \end{aligned}.$$

The price of on-demand and daily billing is fixed, but the price of auction is dynamically adjusted, and the overbooking ratio is also determined online. Details of the online auction approach and dynamic overbooking mechanism are described in III and IV, respectively.

III. Online Auction Approach

In this section, we first describe the pricing rule and the winner determination rule of our auction approach in III-A. Then, we theoretically analyze our approach in III-B.

A. Pricing Rule and Winner Determination Rule

A trusted third party called auctioneer is necessary to administer the transaction between mobile tasks and servers. The auctioneer first collects bids from tasks and asking prices from servers. Then it determines the matching of winning buyers and winning sellers, the prices charged to the buyers, and the payment to the sellers. The auction approach should satisfy the following three properties [10], [12]:

- **Individual rationality**: no winning buyer is charged more than its bid, and no winning seller is rewarded less than its asking price.
- **Computational efficiency**: the auction outcome is tractable with polynomial time complexity.
- **Truthfulness**: the bid submitted by each mobile device should be truthful, i.e., no buyer can improve its utility by submitting a bid different from its true valuation.

We design the online auction approach based on McAfee's mechanism [26]. McAfee's mechanism achieves individual

rationality and truthfulness, but neither balanced budget nor economic efficiency. In McAfee's mechanism, one seller can only accept one buyer, which is not suitable in FC scenario, while our auction approach consisting of a pricing rule and a winner determination rule can support one seller trading with multiple buyers. Similar to McAfee's mechanism, our auction approach is individually rational and truthful, and the theoretical analysis is shown in III-B. The pseudocode of the pricing rule and winner determination rule can be found in Algorithm 1 and Algorithm 2, respectively. The pricing rule determines the candidate assignments among the servers and the tasks with the corresponding prices. Then the winner determination rule determines the winning bids for each server from the candidate assignments with the corresponding prices.

In Algorithm 1, auctioneer first sorts the bids and the asking prices. Then auctioneer determines the candidate assignment according to the sorted results and the remaining capacity of each server. After that, the pricing of each task and server is determined. Some notations used in Algorithm 1 are introduced as follows. For each task b_k, its actual bid price is denoted as v'_i. Moreover, $R(t) = \{S_i^j.a(t)|S_i^j \in F\}$ is the available resource for all servers, where $S_i^j.a(t) = \{S_i^j.a(t).c, S_i^j.a(t).m\}$ is the available CPU and memory resource. In addition, $D(t) = \{S_i^j.w(t)|S_i^j \in F\}$ is the set of asking prices of all servers, where $S_i^j.w(t) = \{S_i^j.w(t).c, S_i^j.w(t).m\}$ is the asking price of CPU and memory resource. B_a and S_a are the sets of assigned tasks and assigned servers, respectively. σ is the assignment, e.g., $\sigma(k) = \{i, j\}$ means task b_k is assigned to S_i^j. P_a^b is the payment of buyer and P_a^s is the income of seller. For a clear description, we use $\{c, m\}$ to denote the two types of resource. $x = c$ means the resource is CPU, and $x = m$ means the resource is memory.

As Algorithm 1 shows, servers of each node are grouped to receive bids. For node F_i, V_i and W_i are the set of received bids and servers. As shown in line 4, the received bids and asking prices of each resource type are sorted in descending and ascending order, respectively, and the results are denoted as $\mathcal{V}_i$ and $\mathcal{W}_i$. Then, as shown in line 5 - 15, the candidate assignment is determined. For each task b_k, we define $b_k^{i,j}$ as:

$$b_k^{i,j} = 1\{v_k.c_i \geq S_i^j.w(t).c\} \times 1\{v_k.m_i \geq S_i^j.w(t).m\} \quad (4)$$
$$\times 1\{r_k.c \leq S_i^j.a(t).c\} \times 1\{r_k.m \leq S_i^j.a(t).m\} \times 1\{t_k = t\},$$

where $1\{\cdot\}$ is Iverson bracket, which is equivalent to 1 when condition is satisfied. Otherwise, it is equivalent to 0. As shown in line 7 - 11, if $b_k^{i,j} = 1$, it means the bid of task b_k is larger than the asking price of server S_i^j, and the remaining capacity of S_i^j is also larger than the request resource of b_k for both CPU and memory resource, so that b_k can be assigned to S_i^j. The task b_k and server S_i^j are added to the candidate assignment sets $B_a^{x,i}$ and $S_a^{x,i}$ of F_i with resource type x, respectively. Then the assignment σ is updated, task b_k is removed from the set of sorted tasks $\mathcal{V}_i$, and the available resource $S_i^j.a(t)$ is updated.

After this step, the pricing is determined in line 17 - 30. If there are unassigned bids and servers, the pricing of each resource type $P = \{P.x|x \in \{c, m\}\}$ is calculated based on McAfee's mechanism as [26]:

$$P.x = \frac{v'_{|B_a^{x,i}|+1}.x_i + S_i^{|S_a^{x,i}|+1}.w(t).x}{2}. \quad (5)$$

If $P.x$ is between $S_i^{|S_a^{x,i}|}.w(t).x$ and $v'_{|B_a^{x,i}|}.x_i$, the price charged from bid $P_i^b.x$ and the price rewarded to the server $P_i^s.x$ are set to $P.x$ [26]. Otherwise, we cancel all the assigned bids of the $|S_a^{x,i}|$-th server, and set the price as the highest bid of the $|S_a^{x,i}|$-th server. As shown in line 32, if task b_k is in the candidate assignment sets of F_i which are $B_a^{c,i}$ and $B_a^{m,i}$, it means that b_k meets both the requirements of CPU and memory resource, and we add b_k to the assignment set B_a^i. Finally, we get the output as shown in line 34 - 35.

Based on the pricing rule, the candidate bids with the corresponding prices $P_i^b.x$ and $P_i^s.x$ are determined. We then introduce how to make the winner determination in Algorithm 2. In Algorithm 2, B_w, S_w are the sets of winning tasks and servers, respectively. σ_w is the winning match between tasks and servers. In the winner determination, as shown in line 1 and line 3, auctioneer first sorts servers and bids by a weighted sum of prices of CPU and memory resource, i.e., $p_i^j = \alpha_p \times S_i^j.w(t).c + \beta_p \times S_i^j.w(t).m$, and $p_k = \alpha_p \times v_k.c + \beta_p \times v_k.m$, where α_p and β_p control the weights. Then, the winning bid is determined from the candidate sets as shown in line 4 - 17. If the server S_i^j still has enough resource capacity, task b_k is assigned to it. Besides, the winning match σ_w is updated, the sets of winning tasks B_w and servers S_w are updated with the corresponding prices. After that, as shown in line 9 - 15, b_k is removed from all other candidate sets, and the $S_i^j.a(t)$ is updated.

B. Theoretical Analysis

Next, we prove our auction approach holds the properties of computational efficiency, individual rationality, and truthfulness.

Theorem 1. *The proposed auction approach achieves the individual rationality for each bid.*

Proof. In Algorithm 1, there are two cases for task b_k to be assigned as a buyer candidate and for server S_i^j to become a seller candidate.

- $S_i^{|S_a^{x,i}|}.w(t).x < P.x < v'_{|B_a^{x,i}|}.x_i$: In this case, task b_k must have an actual bid price v'_k that $v'_k.x_i \geq v'_{|B_a^{x,i}|}.x_i$ due to the decreasing order in the set of sorted tasks $\mathcal{V}_i$. So that $v'_k.x_i \geq v'_{|B_a^{x,i}|}.x_i > P.x$. Moreover, the server must have an asking price $S_i^j.w(t).x \leq S_i^{|S_a^{x,i}|}.w(t).x$ due to the increasing order in the set of sorted servers $\mathcal{W}_i$, so each asking price of servers satisfies that $S_i^j.w(t).x \leq S_i^{|S_a^{x,i}|}.w(t).x < P.x$.
- Otherwise, $P.x \notin [S_i^{|S_a^{x,i}|}.w(t).x, v'_{|B_a^{x,i}|}.x_i]$. In this case, the price is set to $\max\{v'_k.x_i|b_k \in B_a^p\}$. For each task

Algorithm 1 Pricing Rule

Input: $B, R(t), D(t)$
Output: B_a, S_a, P_a^b, P_a^s

Set $B_a, S_a, \sigma, P_a^b, P_a^s = \emptyset$
for $F_i \in F$ **do**
  for $x \in \{c, m\}$ **do**
    Sort $V_i = \{b_k\}$ to $\mathcal{V}_i$ and $W_i = \{S_i^j\}$ to $\mathcal{W}_i$
    for $S_i^j \in \mathcal{W}_i$ **do**
      for $b_k \in \mathcal{V}_i$ **do**
        if $b_k^{\{i,j\}} = 1$ **then**
          $B_a^{x,i} = B_a^{x,i} \bigcup \{b_k\}, S_a^{x,i} = S_a^{x,i} \bigcup \{S_i^j\}$
          $\sigma(k) = \{i, j\}, \mathcal{V}_i = \mathcal{V}_i / \{b_k\}$
          $S_i^j.a(t).x = S_i^j.a(t).x - r_k.x$
        else
          break
        end if
      end for
    end for
    $P = 0$
    if $|B_a^{x,i}| < |\mathcal{V}_i|$ and $|S_a^{x,i}| < |\mathcal{W}_i|$ **then**
      Calculate $P.x$ by Eq. (5)
    end if
    if $S_i^{|S_a^{x,i}|}.w(t).x < P.x < v'_{|B_a^{x,i}|}.x_i$ **then**
      $P_i^b.x = P_i^s.x = P.x$
    else
      for $b_k \in B_a^{x,i}$ **do**
        if $\sigma(k) = \{i, |S_a^{x,i}|\}$ **then**
          $B_a^{x,i} = B_a^{x,i} / \{b_k\}, B_a^p = B_a^p \bigcup \{b_k\}$
        end if
      end for
      $S_a^{x,i} = S_a^{x,i} / \{S_i^{|S_a^{x,i}|}\}$
      $P_i^b.x = P_i^s.x = \max\{v'_k.x_i | b_k \in B_a^p\}$
    end if
  end for
  $B_a^i = B_a^{c,i} \bigcap B_a^{m,i}, S_a^i = S_a^{c,i} \bigcap S_a^{m,i}$
end for
$B_a = B_a^1 \bigcap B_a^2 ... \bigcap B_a^m$, $S_a = S_a^1 \bigcap S_a^2 ... \bigcap S_a^m$
$P_a^b = \{P_i^b | F_i \in F\}, P_a^s = \{P_i^s | F_i \in F\}$
end

b_k, we can easily get $v'_k.x_i \geq v'_{|B_a^{x,i}|}.x_i > P.x$. And for each server S_i^j, we can get $S_i^j.w(t).x \leq S_i^{|S_a^{x,i}|}.w(t).x$. In addition, it is obvious that $S_i^{|S_a^{x,i}|}.w(t).x \leq \max\{v'_k.x_i | b_k \in B_a^p\}$. So $S_i^j.w(t).x \leq \max\{v'_k.x_i | b_k \in B_a^p\}$.

Therefore, each buyer assigned in Algorithm 1 is never charged a price higher than its bid, while each seller assigned is rewarded a payment not less than its asking price, which ensures individual rationality for both buyers and sellers. □

Theorem 2. *The proposed auction approach is computationally efficient.*

Proof. To analyze the time complexity, we use l to denote the total number of servers. To implement Algorithm 1, for each node and each resource type, we first sort the tasks and the servers with a time complexity $O(n \log n)$ and $O(l \log l)$, respectively. Then, within the for-loop for the candidate assignment, the time complexity is $O(ln)$. The time complexities of obtaining $P.x$ and pricing are $O(1)$ and $O(n)$, respectively. In total, the time complexity of Algorithm 1 is $O(mn \log n) + O(ml \log l) + O(mln)$.

Algorithm 2 Winner Determination Rule

Input: $B, R(t), D(t), B_a, S_a, P_a^b, P_a^s$
Output: $B_w, S_w, \sigma_w, P_w^b, P_w^s$

Sort S_a to $\mathcal{S}$ in increasing order of p_i^j
for $S_i^j \in \mathcal{S}$ **do**
  Sort B_a^i to $\mathcal{B}_a^i$ in decreasing order of p_k
  for $b_k \in B_a^i$ **do**
    if $S_i^j.a(t).c \geq r_k.c$ and $S_i^j.a(t).m \geq r_k.m$ **then**
      $\sigma_w(k) = \{i, j\}, \sigma_w = \sigma_w \bigcap \{\sigma_w(k)\}$
      $B_w = B_w \bigcap \{b_k\}, S_w = S_w \bigcap \{S_i^j\}$
      $P_w^b = P_w^b \bigcap \{P_i^b\}, P_w^s = P_w^s \bigcap \{P_i^s\}$
      for $y \in \{1, 2, .., m\}$ **do**
        if $b_k \in B_a^y$ **then**
          $B_a^y = B_a^y / \{b_k\}$
        end if
      end for
      $S_i^j.a(t).c = S_i^j.a(t).c - r_k.c$
      $S_i^j.a(t).m = S_i^j.a(t).m - r_k.m$
    end if
  end for
end for
end

In Algorithm 2, the servers are sorted with time complexity $O(l \log l)$. Then, within the first for-loop, the bids are sorted with time complexity $O(n \log n)$, and the time complexity of winning assignment is $O(mn)$. In total, Algorithm 2 has a time complexity of $O(l(\log l + n \log n + mn))$. Therefore, the overall time complexity is polynomial. □

To demonstrate the truthfulness, we should prove that each mobile task will honestly submit all of its real costs. The proposed mechanism is truthful if and only if the following two conditions are satisfied [12], [27]: 1) the winning bids determination algorithm is monotonic, and 2) each winning bid pays the critical value. The definitions of monotonicity and critical value are described as follows:

Definition 1. Monotonicity: for each task b_{k_1}, if b_{k_1} wins, then b_{k_2} also wins, where the corresponding bids of b_{k_1} and b_{k_2} are v'_{k_1} and $v'_{k_2} = v'_{k_1} + \sigma(\sigma > 0)$, respectively.

Definition 2. Critical Value: For each task b_k, there is a critical value P_k^b. If the bid of b_k declares a cost that is not larger than P_k^b, it must win. Otherwise, it will lose.

Lemma 1. *The winner determination process in Algorithm 2 is monotonic.*

Proof. Assume that b_{k_i} is one of the winning tasks determined

in the i-th step of Algorithm 2, which means $i-1$ tasks have won in the previous $i-1$ steps. Let $(b_{k_1}, b_{k_2}, ..., b_{k_i})$ be the list of the winning tasks that have been determined in the first i steps. If b_{k_i} was replaced by another task, e.g., b_{k_j}, where the corresponding bids of b_{k_i} and b_{k_j} are v'_{k_i} and $v'_{k_j} = v'_{k_i} + \sigma(\sigma > 0)$, respectively. According to Algorithm 2, b_{k_j} must win in the i-th step or even earlier step. As a result, our auction approach is monotonic. □

Lemma 2. *The winning bid in Algorithm 2 pays the critical value.*

Proof. We assume that task b_{k_l} wins its bid for server S_i^j in the l-th step of Algorithm 2. In this case, the payment of b_{k_l} is set to P_i^b. For $\gamma > 0$, another bid with submitted price $v'_{k_l} = P_i^b + \gamma$ would win, because its cost per unit resource must be higher than the valuation of b_{k_l}. But the bid $v'_{k_l} = P_i^b - \gamma$ will not win, as its valuation must be lower than the valuation of task b_{k_l}. Hence, we prove the above lemma. □

According to the above analysis, we can easily get the following theorem through Lemmas 1 and 2 [12], [27]. Hence, the theorem is proved.

Theorem 3. *The proposed auction approach is truthful.*

The auction approach solves the pricing of the resource of auction billing while guaranteeing the individual rationality, computational efficiency, and truthfulness. Based on the pricing model consisting of auction and two other billing methods, we can overbook the resource to achieve more profit. The dynamic overbooking mechanism is described in the next section.

IV. Dynamic Overbooking Mechanism

In this section, we present the dynamic overbooking mechanism based on resource utilization prediction and SLAV feedback. The resource utilization is a time-sequential sequence, which can be predicted with long short-term memory (LSTM) based neural network [18]. We present the resource utilization prediction with a deep neural network and the dynamic overbooking mechanism in IV-A and IV-B, respectively.

A. Resource Utilization Prediction

To dynamically overbook the resource, we need to predict the resource utilization for next time slot. In other words, we need an effective resource utilization prediction method.

The LSTM based neural network architecture used in our paper is composed of one input layer, two LSTM layers, one linear layer and one output layer. The input of our neural network model includes CPU and memory resource utilization, server attributes, and task information. The output is the CPU and memory resource utilization of each server for the next time slot. We take symmetric mean absolute percentage error (SMAPE) as loss function, which is defined as follows [28]:

$$SMAPE = \frac{1}{n}\sum_{t=1}^{n}\frac{|A'_t - A_t|}{(|A_t| + |A'_t|)/2},$$

where A_t is the actual value, A'_t is the corresponding predicted value. The Adam optimization function is used to optimize the neural network, which is defined as:

$$cache^t = a \times cache^{\alpha_c - 1} + (1 - \alpha_c)(g^t)^2,$$
$$\omega^{t+1} = w^t - \frac{\eta}{\sqrt{cache^t + \epsilon}} g^t,$$

where ω^t is the learning rate at t, ϵ is a smoothing parameter, which is usually set to $[10^{-8}, 10^{-4}]$ to avoid the denominator to become 0, and α_c is the decay rate.

Algorithm 3 Dynamic Overbooking

Input: $B, R(t), D(t), L(0) = \{1\}$
Output: R

```
1:  for t ∈ [1, T] do
2:    for S_i^j ∈ F do
3:      Get S_i^j.p(t) from neural network
4:      Calculate S_i^j.a(t) by Eq. (6)
5:      if L_i^j(t − 1) < 1 then
6:        S_i^j.a'(t) = 0
7:      else
8:        Calculate S_i^j.a'(t) by Eq. (7)
9:      end if
10:     Calculate S_i^j.w(t) by Eq. (8)
11:   end for
12:   Send B, D(t), R'(t) to auctioneer
13:   {B_a, S_a, P_a^b, P_a^s} = PricingRule(B, D(t), R'(t))
14:   {B_w, S_w, σ_w, P_w^b, P_w^s} = WinnerDetermination
        Rule(B, R'(t), D(t), B_a, S_a, P_a^b, P_a^s)
15:   Calculate L(t) = {L_i^j(t) | S_i^j ∈ F} by Eq. (1)
16:   Calculate R by Eq. (3)
17: end for
18: end
```

B. Dynamic Overbooking Mechanism

The design of dynamic overbooking mechanism is to maximize the profit of the nodes through overbooking with a high degree of QoS satisfaction for the on-demand and daily tasks, which is described as follows.

The procedure of the dynamic overbooking mechanism is shown in Algorithm 3. For each time slot t, the servers first get the predicted resource utilization from the neural network and calculate the available resource for overbooking. Then the servers adjust the available resource according to the SLAV feedback and update the asking price. After that, auctioneer collects necessary information and calls Algorithms 1 and 2. Finally, the SLAV is updated, and the profit is calculated.

First, as shown in line 3 - 4, the predicted CPU and memory resource utilization of each server $S_i^j.p(t) = \{S_i^j.p(t).c, S_i^j.p(t).m\}$ is obtained from the neural network. Then, the available CPU and memory resource $S_i^j.a(t) = \{S_i^j.a(t).c, S_i^j.a(t).m\}$ is obtained as:

$$S_i^j.a(t).x = S_i^j.a_0.x - S_i^j.p(t).x - S_i^j.s(t).x, \quad (6)$$

where $x \in \{c, m\}$ is the type of resource, $S_i^j.a_0 = \{S_i^j.a_0.c, S_i^j.a_0.m\}$ is the resource capacity of server S_i^j, and $S_i^j.s(t) = \{S_i^j.s(t).c, S_i^j.s(t).m\}$ is the total resource rented through the auction, which is obtained as:

$$S_i^j.s(t).x = \sum_{\{b_k|t_k^s \le t \le t_k^{e\prime}\} \bigcap \{b_k|\sigma(k)=\{i,j\}\}} r_k.x,$$

where t_k^s is the estimated start time of task b_k. $t_k^{e\prime}$ is the end time according to the auction billing that the auction task can last at most T_a time slot, which is defined as:

$$t_k^{e\prime} = \begin{cases} t_k^s + |T_a|, & t_k^e - t_k^s > |T_a| \\ t_k^e, & t_k^e - t_k^s \le |T_a| \end{cases}.$$

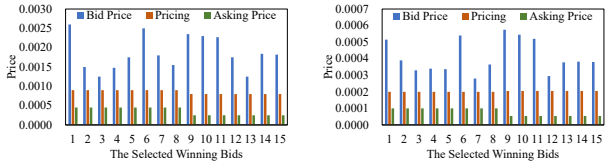
Then, as shown in line 5 - 10, the available resource for overbooking is adjusted and the asking price is updated. To avoid excessive SLAV, we adopt a TCP congestion control like dynamic available resource adjustment [19]. When the degree of QoS satisfaction $L_i^j(t-1) < 1$, it means there is SLAV for server S_i^j, and the available resource for auction billing $S_i^j.a'(t)$ of server S_i^j is set to zero. Otherwise, $S_i^j.a'(t)$ is obtained as:

$$S_i^j.a'(t) = \begin{cases} \min\{(S_i^j.a'(t-1)+\epsilon) \times \alpha_s, S_i^j.a(t)\}, \\ \qquad S_i^j.a(t) > th_{over} \\ \min\{S_i^j.a'(t-1) + \beta_s, S_i^j.a(t)\}, \\ \qquad th_{under} < S_i^j.a(t) < th_{over} \\ \min\{S_i^j.a'(t-1), S_i^j.a(t)\}, \\ \qquad S_i^j.a(t) < th_{under} \end{cases}, \tag{7}$$

where ϵ is a small positive constant that ensures $S_i^j.a'(t) \neq 0$, α_s and β_s are the parameters controlling the increment speed of available resource. Moreover, th_{over} and th_{under} are the over and under threshold of available resource, respectively. Eq. (7) is applied for both CPU and memory resource. When there is a SLAV, the available resource for auction is set to zero. Then the available resource is updated according to the equation. After that, the asking price $S_i^j.w(t)$ is calculated by Eq. (8). The asking price should be larger than the cost price $S_i^j.w_0$ of each server. $S_i^j.w(t)$ is obtained as:

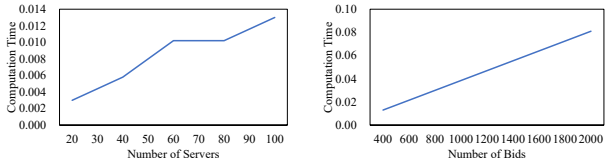
$$S_i^j.w(t).x = \begin{cases} \frac{1}{L_i^j(t-1)} \times \frac{1-S_i^j.a(t).x}{1-th_{over}.x} \times S_i^j.w_0.x, \\ \qquad S_i^j.a(t).x > th_{over}.x \\ \frac{1}{L_i^j(t-1)} \times \frac{1-S_i^j.a(t).x}{1-th_{under}.x} \times S_i^j.w_0.x, \\ \qquad S_i^j.a(t).x < th_{under}.x \\ \frac{1}{L_i^j(t-1)} \times S_i^j.w_0.x, \qquad \text{Otherwise} \end{cases}. \tag{8}$$

After the update of the available resource and the asking price, as shown in line 12 - 14, auctioneer collects the set of available resource $R'(t) = \{S_i^j.a'(t) | S_i^j \in F\}$, the set of asking prices $D(t)$ of the servers, and the set of tasks B. Then auctioneer performs the auction by calling Algorithms 1 and 2. Finally, in line 15 - 16, the degree of QoS satisfaction $L(t)$ is updated and will be used for next time slot. The profit is calculated.



(a) CPU (b) Memory

Fig. 1. Individual Rationality



(a) Different Number of Servers (b) Different Number of Bids

Fig. 2. Computational Efficiency

V. Performance Evaluation

In this section, we first describe the experimental settings in V-A. Then, the performance of the auction approach and the dynamic overbooking mechanism is illustrated in V-B and V-C, respectively.

A. Experimental Settings

The data set we've used in our experiment is chosen from the Google cluster trace [21], [22], and the initial asking prices are set according to Amazon Web Services (AWS) [23].

Auction approach: it is assumed that there exist 100 servers and these servers are randomly assigned to 5 nodes. These 100 servers can be divided into six types according to their resource capacity, which shows the heterogeneity in FC. 1×10^6 tasks are chosen from the data set as mobile tasks for the auction. During each time slot of 5 minutes, there are about 2000 tasks in each auction. For each task b_i, its start time t_i^s, end time t_i^e, and request r_i are extracted from the data set.

Dynamic overbooking mechanism: 6×10^7 tasks are chosen from the date set and separated into the training set and test set independently to train and validate the neural network predictor. Besides, the tasks that run on these servers are separated into on-demand and daily tasks according to their priority. According to Eq. (2), the unit prices of on-demand and daily billing are set to \$0.0107 and \$0.0038 per hour for CPU and memory, respectively. Furthermore, the power consumed by one CPU unit h_c and memory unit h_m are set as 0.008 and 0.00014, respectively, and the unit power fare e is \$0.2 [25]. In addition, $\alpha_p = \beta_p = 1$, $th_{under} = 0.1$, $th_{over} = 0.55$, $\alpha_s = 2$, $\beta_s = 0.05$.

B. Auction

We will illustrate the performance of our auction approach in the following three aspects: individual rationality, computational efficiency, and truthfulness.

Individual rationality: the bid price, asking price, and the pricing of some winning tasks are shown in Fig. 1. In Fig. 1,

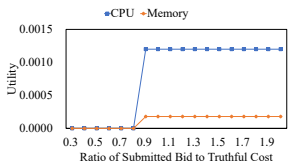


Fig. 3. Truthfulness

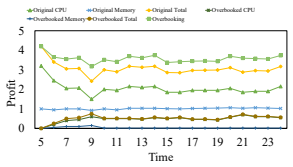


Fig. 4. Average Profit of the Node

each winning task is charged a price not higher than its bid price, while each winning server receives a payment not less than its asking price. Therefore, the proposed online auction approach achieves individual rationality.

Computational efficiency: the tests of computational efficiency are run on a Linux Server with 2.50 GHz Intel Xeon CPU E5-2609 and 16 GB memory. The computation time with different number of servers and bids are shown in Fig. 2(a) and Fig. 2(b), respectively. From Fig. 2, we can see that the proposed auction approach is subject to polynomial computation time concerning the number of servers and bids.

Truthfulness: as shown in Fig. 3, the value in x-axis is defined as the ratio of the submitted price to the truthful valuation, i.e., $\frac{v_i'.c}{v_i.c}$ for the CPU resource. The value in y-axis is the utility, which is defined as the valuation v_i minus the pricing P_i. When the ratio is 1, the submitted price equals to the truthful valuation $v_i.c$. From Fig. 3, we can conclude that the maximum utility is achieved when the task submits the truthful information, i.e., $\frac{v_i'.c}{v_i.c} = 1$. As a result, the task cannot improve its utility through any other bids, which guarantees the truthfulness.

C. Dynamic Overbooking

In this sub-section, we describe the performance of our dynamic overbooking mechanism. First, we validate the accuracy rate of our neural network. Then the SLAV and profit of our dynamic overbooking mechanism are discussed.

To effectively demonstrate the accuracy rate of predicted resource utilization of our network, we use final state-based (FS) method, simple moving average (SMA) method, and exponential moving average (EMA) method as our baselines [29]. In FS method, the information of tasks during last time slot is used to predict, while the information of tasks during n_w last time slot is used in SMA method, where n_w is the windows size. In EMA method, the prediction is based on the weighted sum of the previous tasks, which is obtained as:

$$Pre(t) = \alpha_r \times J_1 + (1 - \alpha_r) \times Pre(t-1),$$

where $Pre(t)$ is the predicted value at time slot t, J_1 is the value of the tasks during the last time slot, α_r is the decay parameter to optimize the accuracy and adjusted by experience. The results of predicted resource utilization of different methods are shown in TABLE I. We can conclude that the accuracy rate of our proposed LSTM based neural network is much higher than the baselines.

The performance of a randomly selected node and server is shown in Fig. 5 and Fig. 6, respectively. Fig. 5(a) shows the

TABLE I
THE PREDICTION RESULTS

Prediction Method	Accuracy Rate
Final State-based Method	0.674
Simple Moving Average Method (n_w = 5)	0.753
Exponential Moving Average Method ($\alpha_r = 0.9$)	0.645
Exponential Moving Average Method ($\alpha_r = 0.95$)	0.657
Exponential Moving Average Method ($\alpha_r = 1$)	0.680
Proposed LSTM based Neural Network	**0.815**

average degree of QoS satisfaction of this node. There is a relatively low degree of QoS satisfaction at the beginning, but such degree is improved for the available resource adjustment of dynamic overbooking mechanism. The average price of the node is shown in Fig. 5(b), where the price is dynamically adjusted according to the degree of QoS satisfaction. The CPU and memory utilization are shown in Fig. 5(c) and Fig. 5(d) respectively. It's easy to see that predicted utilization matches well with real utilization, and overall utilization is greatly improved.

The performance of the server which belongs to this node is shown in Fig. 6. First, the degree of QoS satisfaction is demonstrated in Fig. 6(a), where the degree of QoS satisfaction remains high. The price of CPU and memory is shown in Fig. 6(b). There is little change of the price in Fig. 6(b) due to the available resource capacity. In Fig. 6(c) and Fig. 6(d), CPU utilization and memory utilization are improved while the predicted results are achieving a high accuracy rate.

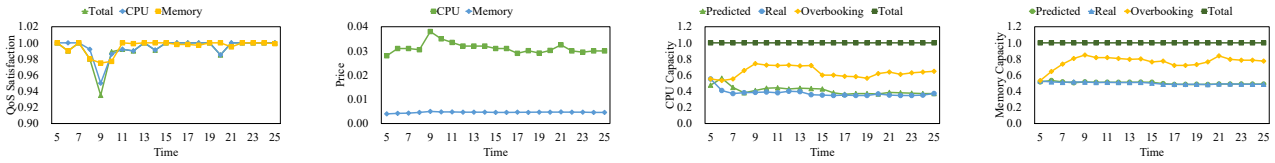
The average profit of the node is shown in Fig. 4. The original CPU and memory profit is the profit without overbooking. And the overbooked CPU and memory profit is the profit of overbooked resource. The total overbooking profit is the sum of the total original profit and the total overbooked profit. From Fig. 4, we can draw the conclusion that through overbooking, the total profit of the node can be greatly improved. To sum up, the dynamic resource overbooking mechanism effectively overbooks the resource with a high degree of QoS satisfaction.

VI. CONCLUSION

In this paper, we have modeled the dynamic resource overbooking mechanism in FC. We first describe our system model, pricing model and problem formulation of overbooking problem. Secondly, we propose an online auction approach and prove its individual rationality, computational efficiency, and truthfulness. Thirdly, we describe the dynamic overbooking mechanism based on resource utilization prediction and SLAV feedback. We conduct the experiments with real-world data trace, and the experimental results show that our auction approach and dynamic overbooking mechanism are efficient. Future work will consider the resource overbooking across fog nodes and cloud data centers.

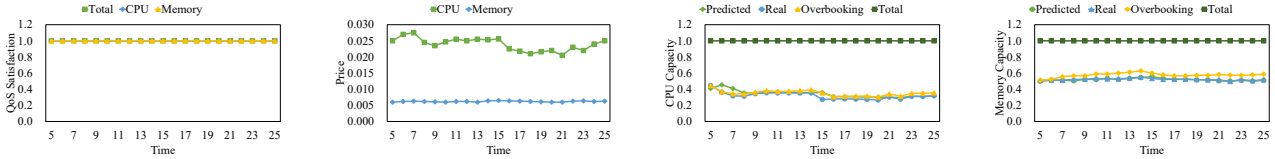
ACKNOWLEDGMENT

This work is supported by FDCT/0007/2018/A1, DCT-MoST Joint-project No. (025/2015/AMJ), University of Macau Grant Nos: CPG2018-00032-FST & SRG2018-00111-FST of



(a) Degree of QoS Satisfaction (b) Price (c) CPU Utilization (d) Memory Utilization

Fig. 5. Performance of the overbooking mechanism of the Node



(a) Degree of QoS Satisfaction (b) Price (c) CPU Utilization (d) Memory Utilization

Fig. 6. Performance of the overbooking mechanism of the Server

SAR Macau, China; Chinese National Research Fund (NSFC) Key Project No. 61532013 and National China 973 Project No. 2015CB352401.

REFERENCES

[1] J. Li, J. Jin, D. Yuan, M. Palaniswami, and K. Moessner, "Ehopes: Data-centered fog platform for smart living," in *IEEE International Telecommunication Networks and Applications Conference (ITNAC)*, 2015, pp. 308–313.

[2] Z. Tang, X. Zhou, F. Zhang, W. Jia, and W. Zhao, "Migration modeling and learning algorithms for containers in fog computing," *IEEE Transactions on Services Computing*, 2018.

[3] Y.-H. Kao, B. Krishnamachari, M.-R. Ra, and F. Bai, "Hermes: Latency optimal task assignment for resource-constrained mobile computing," *Transactions on Mobile Computing*, vol. 16, no. 11, pp. 3056–3069, 2017.

[4] R. Deng, R. Lu, C. Lai, T. H. Luan, and H. Liang, "Optimal workload allocation in fog-cloud computing toward balanced delay and power consumption," *IEEE Internet of Things Journal*, vol. 3, no. 6, pp. 1171–1181, 2016.

[5] O. C. A. W. Group *et al.*, "Openfog reference architecture for fog computing," *OPFRA001*, vol. 20817, no. 1, pp. 1–162, 2017.

[6] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, and S. Chen, "Vehicular fog computing: A viewpoint of vehicles as the infrastructures," *Transactions on Vehicular Technology*, vol. 65, no. 6, pp. 3860–3873, 2016.

[7] L. F. Bittencourt, M. M. Lopes, I. Petri, and O. F. Rana, "Towards virtual machine migration in fog computing," in *IEEE International Conference on P2P, Parallel, Grid, Cloud and Internet Computing (3PGCIC)*, 2015, pp. 1–8.

[8] H. Zhang, Y. Xiao, S. Bu, D. Niyato, R. Yu, and Z. Han, "Fog computing in multi-tier data center networks: A hierarchical game approach," in *IEEE International Conference on Communications (ICC)*, 2016, pp. 1–6.

[9] V. Krishna, *Auction theory*. Academic press, 2009, vol. 1.

[10] A.-L. 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*, vol. 9, no. 6, pp. 895–909, 2016.

[11] A.-L. Jin, W. Song, and W. Zhuang, "Auction-based resource allocation for sharing cloudlets in mobile cloud computing," *IEEE Transactions on Emerging Topics in Computing*, vol. 6, no. 1, pp. 45–57, 2018.

[12] X. Wang, X. Chen, and W. Wu, "Towards truthful auction mechanisms for task assignment in mobile device clouds," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[13] I. S. Moreno and J. Xu, "Neural network-based overallocation for improved energy-efficiency in real-time cloud environments," in *IEEE International Symposium on Object/Component/Service-Oriented Real-Time Distributed Computing (ISORC)*, 2012, pp. 119–126.

[14] ——, "Customer-aware resource overallocation to improve energy efficiency in realtime cloud computing data centers," in *IEEE International Conference on Service-Oriented Computing and Applications (SOCA)*, 2011, pp. 1–8.

[15] S. Barbarossa, E. Ceci, and M. Merluzzi, "Overbooking radio and computation resources in mmw-mobile edge computing to reduce vulnerability to channel intermittency," in *Networks and Communications (EuCNC), 2017 European Conference on*. IEEE, 2017, pp. 1–5.

[16] F. Slim, F. Guillemin, and Y. Hadjadj-Aoul, "Close: A costless service offloading strategy for distributed edge cloud," in *Consumer Communications & Networking Conference (CCNC)*. IEEE, 2018, pp. 1–6.

[17] M. T. Imam, S. F. Miskhat, R. M. Rahman, and M. A. Amin, "Neural network and regression based processor load prediction for efficient scaling of grid and cloud resources," in *IEEE International Conference on Computer and Information Technology (CIT)*, 2011, pp. 333–338.

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

[19] M. Allman, V. Paxson, and E. Blanton, "TCP Congestion Control," RFC 5681, September 2009.

[20] N. Raveendran, H. Zhang, Z. Zheng, L. Song, and Z. Han, "Large-scale fog computing optimization using equilibrium problem with equilibrium constraints," in *GLOBECOM 2017-2017 IEEE Global Communications Conference*. IEEE, 2017, pp. 1–6.

[21] J. Wilkes, "More Google cluster data," Google research blog, Nov. 2011, posted at http://googleresearch.blogspot.com/2011/11/more-google-cluster-data.html.

[22] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format + schema," Google Inc., Mountain View, CA, USA, Technical Report, Nov. 2011, revised 2014-11-17 for version 2.1. Posted at https://github.com/google/cluster-data.

[23] A. W. Services. (2018) Aws cloud pricing principles. [Online]. Available: https://aws.amazon.com/pricing/

[24] M. Azure. (2017) Sla for virtual machines. [Online]. Available: https://azure.microsoft.com/en-us/support/legal/sla

[25] J. Li, Y. Zhu, J. Yu, C. Long, G. Xue, and S. Qian, "Online auction for iaas clouds: Towards elastic user demands and weighted heterogeneous vms," in *IEEE International Conference on Computer Communications (INFOCOM)*, 2017, pp. 1–9.

[26] R. P. McAfee, "Mechanism design by competing sellers," *Econometrica: Journal of the econometric society*, vol. 1, no. 1, pp. 1281–1312, 1993.

[27] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic game theory*. Cambridge University Press Cambridge, 2007, vol. 1.

[28] P. Kar, "Workload prediction in cloud datacenters based on user behavior modeling," Ph.D. dissertation, Birla Institute of Technology and Science, Pilani, 2016.

[29] S. Zhao, H. Chen, R. Zhao, Y. Zhao, and G. Chen, "A big data processing-oriented prediction method of cloud computing service request," *Journal of Applied Science and Engineering*, vol. 19, no. 4, pp. 497–504, 2016.