This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2642182, IEEE Transactions on Services Computing

Distributed Multi-dimensional Pricing for Efficient Application Offloading in Mobile Cloud Computing

Kun Xie[1,2], *IEEE, member*, Xin Wang[2], *IEEE, member*, Gaogang Xie[3], *IEEE, member*
Dongliang Xie[2], *IEEE, member*, Jiannong Cao[4], *IEEE, fellow*, Yuqin Ji[2], Jigang Wen[3]
[1] College of Computer Science and Electronics Engineering, Hunan University, Changsha, 410082, China
[2] Department of Electrical and Computer Engineering, State University of New York at Stony Brook, USA
[3] Institute of computing technology, Chinese Academy of Science, China
[4]Department of Computing, The Hong Kong Polytechnic University, Hong Kong
kun.xie@stonybrook.edu, x.wang@stonybrook.edu, xie@ict.ac.cn,
csjcao@comp.polyu.edu.hk, dongliang.xie@stonybrook.edu, yuqinji@cnsunet.com, wenjigang@ict.ac.cn

***Abstract*—Offloading computation intensive applications to mobile cloud is promising for overcoming the problems of limited computational resources and energy of mobile devices. However, without considering the competition relationship of mobile users and cloudlets in the mobile cloud computing system, existing studies lack an incentive mechanism for the system to achieve efficient application offloading and cloud resource provisioning. In this paper, we design MPTMG, a Multi-dimensional Pricing mechanism based on Two-sided Market Game. We propose three types of prices: a multi-dimensional price corresponding to multi-dimensional resource allocation, a penalty price to encourage fair and high quality cloud services, and a benefit discount factor to motivate more even provisioning of resources on different dimensions in the cloud. Based on these prices, we propose a distributed price-adjustment algorithm for efficient resource allocation and QoS-aware offloading scheduling. We prove that the algorithm can converge in a finite number of iterations to the equilibrium core allocation at which the mobile cloud system achieves the Pareto efficiency by maximizing the total system benefit. To the best of our knowledge, this is the first paper that applies economic theories and pricing mechanisms to manage application offloading in mobile cloud systems. The simulation results demonstrate that our proposed pricing mechanism can significantly improve the system performance.**

I. Introduction

Mobile cloud computing system, an emerging mobile computing paradigm, has received a lot of recent interests [1]. To alleviate the constraints of computational resources and energy of mobile devices, many efforts have been made in recent years to consider various ways of offloading computation-intensive applications to a powerful cloud. For examples, Google Voice Search and Apple Siri [2], perform compute intensive speech recognition by exploiting the resources in the cloud, which presents the rich commercial opportunities. With the technique developments on context-aware scene interpellation [3] and the support of cloud resources, we can imagine the emerge of new interactive and resource-intensive mobile applications such as speech and image recognition, natural language processing, situational awareness. These applications will enrich our daily lives with functionalities that go beyond what today's Google Voice Search and Apple Siri can offer.

Despite the potential of offloading computationally expensive tasks to the powerful cloud to enable mobile devices to conserve energy and run advanced software and applications, the recent study in [4] reports an average RTT of 74 milliseconds from 260 global vantage points to the optimal Amazon EC2 instances. Many individual round trips take hundreds to thousands of milliseconds, which makes it difficult to run delay sensitive applications in the remote cloud. The limited wireless network bandwidth and the high latency in wide-area communications pose practical challenges to running delay-sensitive or data-intensive wireless applications in remote cloud servers.

To mitigate these limitations, a new cloud architecture, called cloudlet [3], [5]–[11] is proposed to reduce the communication delay by exploiting the local area networks and cloud servers closeby. As shown in Fig. 1, cloudlets represent the middle tier of a 3 tier hierarchy (mobile user-cloudlet-cloud). Cloudlet is a resource-rich server or server cluster that has the Internet access and is well connected to mobile devices via a high-speed local area network (LAN). Cloudlets may be deployed along with access point (APs) by the same or different operators, and cloud services can be packaged and run in the form of virtual machines (VMs) in the cloudlet. Cloudlet can be viewed as a ”data center in a box” whose goal is to ”bring the cloud closer”. As a powerful, well-connected, one wireless hop away cloud proxy, cloudlets are the enabling technology for a new genre of resource-intensive but latency-sensitive mobile applications. Mobile devices will seek resources from the remote cloud if there are no cloudlets nearby or the local cloudlets do not have space or resources to support the service of a device, at the cost of higher delay.

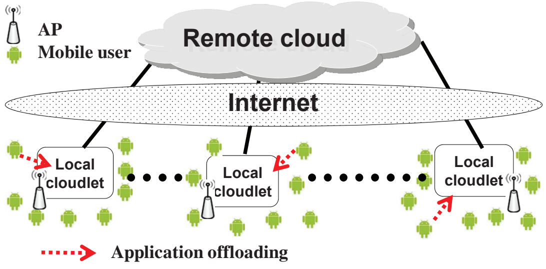


Fig. 1. Cloudlet brings the cloud service closer to users

Although promising, compared to a remote cloud, the resources owned by a cloudlet are much more limited. Therefore, the focus of our work is to achieve efficient resource provisioning in cloudlet-based mobile cloud environment.

Existing efforts on application offloading in mobile cloud systems mainly consider the offloading between a mobile device and a given cloud/cloudlet, with the research focus on application partitioning [12]–[18], prediction [19], [20], and admission control [21]. Despite their importance, existing studies lack an incentive mechanism to motivate mobile users and cloud providers to use the resources more efficiently. Our past studies [22], [23] have proven that economic incentive is very effective in enabling efficient resource usage and resolving many problems that cannot be tackled simply by technical methods. Although some limited efforts have been made to apply economic theories to manage resources in the cloud [24]–[28], over-simplified pricing schemes taken by existing work are far from sufficient for flexible and efficient management of precious resources in mobile clouds. Existing pricing schemes have three major constraints:

- *Pricing based on limited types of VM.* Existing clouds such as Amazon EC2 often charge users based on predefined limited types of VM. Each user application has to be matched to one of the types, and resources are allocated according to the upper limit of the corresponding VM type. As mobile users often have different application requirements thus different resource needs, the allocation based on the upper limit would result in low cloud resource utilization.
- *Single-dimensional pricing.* A cloud application generally requires multiple types of resources, such as CPU, storage, and network bandwidth. Existing clouds charge each VM category a single-dimensional aggregate price. Such a single-dimensional aggregate price is usually set according to the upper limit of the VM category. As mobile users often have different resource requirements, the single-dimensional aggregate price based on the upper limit makes the user pay more than what it needs. On the other hand, although multi-dimensional resource provisioning has been well studied [29]–[31] assuming the full knowledge of user application requests, without considering the cost of each resource type, there is a lack of incentive to motivate users/cloudlets to efficiently request/provision different resources based on their individual loads and limits. The dynamic arrivals of user requests also make conventional multi-dimensional resource provisioning schemes difficult to apply in cloudlets.
- *Unidirectional-charging.* Service charges are often made from the cloud providers (e.g., data center) to users. As a cloudlet often has much lower capacity than a data center, it may be overloaded, which would cause a big delay to the user services. Some mobile users may have higher expectation on the service quality and would only want to pay for the resources and services that satisfy their requirements. Existing pricing schemes fail to encourage the increase of cloud work efficiency and provide users with certain performance support. This may make the mobile cloud system lose its market.

Different from a large cloud which can scale resources more cost-effectively, cloudlets are faced with more constrained resources and dynamic demands as a result of user mobility and limited multiplexing from a smaller number of users. In addition to over simplified pricing, existing VM allocations in cloud systems are either greedy based on instantaneous user requests and available resources without considering their impacts on the long-term performance or optimized assuming the knowledge of all user requests. The unpredictable and dynamic application requests may make resource consumption unbalanced along different dimension within cloudlets, thus reducing the number of applications that can be served by the cloudlets.

To motivate efficient resource usage and ensure higher service performance, we design a Multi-dimensional Pricing (MP) mechanism based on Two-sided Market Game (TMG), and call it MPTMG. The aim of our work is to maximize the benefit of the mobile cloud system through both efficient resource allocation and application offloading scheduling. Specifically, we propose three types of prices: 1) To quantify the benefit from multi-dimensional resource allocation, we apply multi-dimensional price to capture the cost of the cloudlet, and users are allowed to provide multi-dimensional bidding price to compete for different types of resources as needed; 2) To fairly schedule the application offloading and provide better quality support to the applications, a penalty price is charged to the cloudlet when an application request is not handled in time; 3) To maximize resource utilization and accommodate more applications in the presence of unpredictable request arrivals, a benefit discount factor is introduced to encourage more even resource provisioning along different dimensions, which is shown to be effective from our results.

MPTMG is a distributed pricing mechanism, and has the following key features:

1) MPTMG aims to find the optimal resource allocation based on diverse application requirements, and allows mobile users to flexibly require multi-dimensional resources and construct diverse types of VMs. Compared with existing models using fixed types and coarse granularity of VMs, our general resource model helps to significantly improve the efficiency of resource usage in cloudlets.
2) MPTMG can charge mobile users based on their cloudlet resource consumption to meet their application requests. Unlike prior works that only consider a single-dimensional price, to obtain the multi-dimensional price, MPTMG models the mobile cloud system as a VM-cloudlet matching game based on Two-sided Market Game where there are multi-buyers (VMs) and multi-sellers (cloudlets).
3) MPTMG imposes a penalty price (either charged directly or derived from the cost of losing customers) to a cloudlet when it cannot handle a user's request in time, based on which an on-line application-offloading scheduling scheme is applied to ensure better QoS to the applications. The penalty price is levied based on the application priority and waiting time, which serves as an incentive for the cloudlets to improve their service performance while ensuring some fairness and quality to application offloading.
4) MPTMG provides a benefit-discount factor to promote load balanced resource allocation in cloudlets to accommodate more VMs. This in turn reduces the waiting time and allows more delay-sensitive mobile users to obtain the cloud services. With more services supported locally, there will be a large burden reduction for the remote cloud as well as the wide-area networks.
5) MPTMG provides a price adjustment algorithm to enable distributed offloading scheduling and price negotiation among mobile users and cloudlets for a stable and optimal resource allocation. We prove that our algorithm allows the reach of the core allocation in the game where no cloudlet or VM is able to negotiate an agreement that is mutually

more beneficial compared with that of the current, and thus guarantees the cloud system to achieve the Pareto efficiency with the maximum total system utility.

The remainder of this paper is organized as follows. Section II gives an overview of the related work. We introduce our problems and model in Section III, and present utility functions, our price-adjustment algorithm and algorithm analysis in Sections IV, V and VI respectively. We present our simulation results in Section VII and conclude the work in Section VIII.

II. Related Work

We review the literature work from three perspectives: resource provisioning, economic mechanism, and two-side market game.

A. Cloud Offloading and Resource Provisioning

Existing techniques on application offloading in mobile cloud systems mainly focus on application partition [12]–[18], application prediction [19], [20], application admission control [21], and wireless resource allocation for supporting efficient offloading [32], [33]. Compared with offloading a whole application into the cloud, a partitioning scheme is able to achieve a finer-granularity offloading of computational components.

Based on different partitioning algorithms, various offloading scheduling algorithms are proposed [34]–[38] to optimize the network performance. The authors in [34] propose to schedule the computation offloading with a constrained number of cloud resource elements and partition the computation tasks between the mobile side and the cloud side to minimize the average application delay. To minimize the completion time of a user application, for concurrent tasks, a heuristic algorithm [35] is further proposed to offload tasks to the cloud such that the parallelism between the mobile user and the cloud is maximized. In [36], coalesced offloading is proposed to coordinate application offloading requests by sending them in bundles to reduce the period of time that the network interface stays in the high-power state and save the energy of mobile devices. In [37], multiple mobile services in workflows can be invoked to fulfill their complex requirements and determine whether the services of a workflow should be offloaded. In [38], an application-layer protocol, AppATP, is proposed to intelligently manage the mobile-cloud data transmission process to conserve energy.

Existing offloading scheduling algorithms are mostly designed for the traditional mobile cloud with a resource-rich remote data center, while the resources in cloudlets are more constrained.

There are some recent research studies on the wireless resource allocation for mobile-edge cloud computing within the radio access network. The work in [32] considers an MIMO multicell system where multiple mobile users (MUs) ask for computation offloading to a common cloud server, and formulates the offloading problem as the joint optimization of the radio resources and the computational resources to minimize the overall user energy consumption. Authors in [33] study the multi-user computation offloading problem for mobile-edge cloud computing in a multi-channel wireless interference environment. Different from these studies, our work targets to enable efficient application offloading with multi-dimensional pricing to achieve high cloudlet resource utilization and satisfy the user QoS requirements.

Multi-dimensional resources have also been considered in the conventional cloud environment to place VMs [31] with multi-dimension resource requirements in the data center while minimizing the number of servers required, and to migrate VMs [29] across the boundary of servers in the data center to maximize the resource utilization. Authors in [39] propose a dominate resource fairness scheduler that achieves the max-min fairness in the data center by taking into account the heterogeneous demands of applications.

The above multi-dimension resource management schemes generally run centrally, which may be suitable to work in the data center of the traditional cloud systems. However, they cannot work efficiently in a mobile cloud system. Different from these schemes, our multi-dimensional pricing scheme is fully distributed, with a distributed price-adjustment algorithm executed for users and cloudlets to negotiate resource allocations.

B. Economic Mechanism in Cloud Computing

Only some limited and initial studies have been made recently on applying economic mechanism in cloud computing environments. Authors in [24] propose a suite of computationally efficient and truthful auction-style pricing mechanisms, which enable users to fairly compete for resources and cloud providers to increase their overall revenue. In order to support continuous media streaming, the pricing scheme in [25] allows multiple VoD providers to make reservations for bandwidth guarantees from cloud service providers. To maximize the profit, the bidding strategy in [26] enables the cloud service broker to accept job requests from cloud users and leverage the opportunistic yet less expensive spot instances for computation in order. The authors in [27] present an infinite horizon revenue maximization framework to tackle the dynamic pricing problem in an infrastructure cloud based on usage time. With the reservation plan, the consumer can reduce the total resource provisioning cost. In [28], authors propose an optimal cloud resource provisioning (OCRP) algorithm to allocate computing resources used in multiple provisioning stages as well as support a long-term plan, e.g., four stages in a quarter plan and twelve stages in a yearly plan. The demand and price uncertainty is considered in OCRP.

Designed to work in a traditional data center environment, the above mechanisms only provide uni-directional charge and single-dimensional price for a limited number of VM types, and cannot ensure efficient resource utilization in mobile cloud systems which generally run diverse and dynamic applications with heterogeneous and multi-dimensional resource requirements.

Recently, to alleviate the interference among mobile users which simultaneously offload their computation tasks to the cloud via the wireless access, Chen *et al.* in [40] formulate the problem of computation offloading decision making among mobile users as a decentralized computation offloading game and design a decentralized offloading mechanism to achieve the Nash equilibrium. Although this is a game-based approach for mobile cloud, the problem is very different from ours.

To maximize the benefit of the whole system while guaranteeing the applications' QoS requirements, this paper proposes an efficient multi-dimensional pricing mechanism based on Two-sided Market Game and a distributed price-adjustment algorithm for efficient resource negotiation.

C. Two side Market Game

Our main focus in this paper is to find the match between VMs and cloudlets to well utilize the resources in the cloudlets. To achieve it, we model our problem as a VM-Cloudlet matching game (in Section III), which is based on two-side market game. In this subsection, we review some related studies on the game.

One of the main functions of many markets and social processes is to match one kind of agent with another: e.g. students and colleges, workers and firms, marriageable men and women. To study the matching processes, Gale and Shapley [41] introduce two side market game model which has two distinct player groups. If a player from one side of the market can be matched only with a player from the other side, Gale and Shapley proposed that a matching (of students and colleges, or men and women) could be regarded as stable only if it left no pair of players on opposite sides of the market who were not matched to each other but would both prefer to be.

In the market game [42], each player in the game has its own preference to select the player on the other side to form a coalition. Competitions exist among members on the same side and between groups on the two sides. The competitions facilitate the game to reach the stable *coalition* which in turn maximizes the system benefit. A natural application of two-sided market game models is to labor markets [43], [44]. Recently, two side market game has been applied in different fields [45], [46]. Although effective, current studies for specific applications with single dimensional pricing can not be directly applied in our VM-cloudlet matching problem.

To maximize the resource utilization and satisfy the service requirements of applications, in our VM-Cloudlet matching game, we design novel utility functions (preference) taking consideration of the load balancing, multi-dimensional resource management, and response time requirement. Moreover, to well provision the multi-dimensional resource in the mobile cloud computing environment, we design multi-dimension price adjustment algorithms. To the best of our knowledge, we are the first that designs the VM-Cloudlet matching game based on the theory of two-side market game to facilitate cloudlet resource management in the mobile cloud environment.

III. Problem and Models

We consider a mobile cloud system with a set of m cloudlets, $M = \{cloudlet_1, cloudlet_2, \ldots, cloudlet_m\}$. Each $cloudlet_j \in M$ has limited d-dimensional resources, i.e., storage space in the unit of Gigabytes, bandwidth in the unit of Mbps, and computing capability of CPU in the unit of MIPS. We use $\vec{s}_j$ (where $\vec{s}_j \in R^d$ and $\vec{s}_j = \{\vec{s}_j[1], \vec{s}_j[2], \cdots, \vec{s}_j[d]\}$) to denote d-dimensional resource capacity vector of $cloudlet_j$, $\vec{s}_j[1]$ is the first dimension capacity of this cloudlet.

Because of the limited resources of mobile users, after partitioning an application and identifying the application computation component to be offloaded to the cloudlet, a mobile user tends to offload some of her computation components to cloudlets for processing in order to save battery energy and/or improve the computational capability. In this paper, we do not address the issue of application partitions, while existing partitioning methods may be leveraged.

We focus on the efficient management of resources of cloudlets, which can better support delay-sensitive or data intensive applications. The use of the remote data center is not an option for these kinds of applications. For the applications that are not sensitive to the delay, the mobile users may choose to offload their tasks to the remote cloud if the nearby cloudlets are overloaded.

For the convenience of presentation, we use the term application to denote both the whole application and a computation component of the application in this paper.

Each application is associated with a VM request, which arrives dynamically in the mobile cloud system. Between time t and $t + \triangle t$, the set of VM requests is denoted as $N = \{VM_1, VM_2, \ldots, VM_n\}$, with the d-dimensional resource requirement of VM_i being $\vec{v}_i$ (where $\vec{v}_i \in R^d$ and $\vec{v}_i = \{\vec{v}_i[1], \vec{v}_i[2], \cdots, \vec{v}_i[d]\}$). Obviously, the resources requested cannot exceed the limit of the cloudlet, i.e., $\vec{v}_i[k]\} \leq \vec{s}_j[k]\}$ for $1 \leq j \leq m$, $1 \leq i \leq n$, $1 \leq k \leq d$.

Similar to WiFi APs, in a given area there may often exist multiple cloudlets, which may belong to the same or different providers. A mobile user that requests VMs needs to pay cloudlets real money or some credits, while cloudlets will gain benefits by providing resources to users.

Two fundamental questions we will answer in this work are:

1) *which applications will be scheduled to offload to which cloudlet at time t?*

2) *what are the optimal prices charged for mobile users to use resources in cloudlets?*

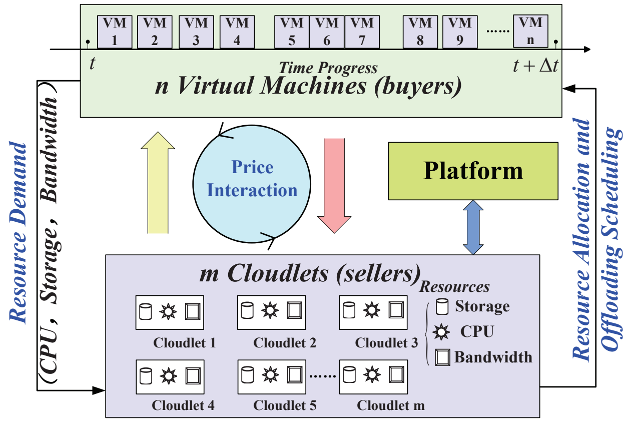


Fig. 2. VM-cloudlet matching game

We consider resource negotiation in mobile cloud systems as a VM-cloudlet matching game (Fig.2). To maximize the resource utilization and satisfy the service requirements of applications, the ***objective*** of this research is to design a pricing mechanism that can maximize the total system benefit. Our model will take into account both resource availability of cloudlets and user VM request dynamics so that both resource provisioning and the sequence of application offloading can be optimally achieved.

In the mobile cloud system, cloudlets and VMs are two distinct player groups. A cloudlet owns the resources and aims to make maximum profit by selling resources to its selected users (each is associated with a VM request) with the basic price set for its multidimensional resources, while a VM is associated with a user which intends to maximize its benefit with the minimum possible payment by selecting the best cloudlet nearby. VMs and cloudlets can obtain the benefit in the game only when the coalitions are formed with the interaction between the buyers and sellers. Each group member sets an appropriate price for a specific resource to maximize its benefit in the coalitions. In our system model, there is a platform to provide users an unified interface to enter the mobile cloud system. Cloudlets need to pay the platform some management fee to enter the market, while the platform will provide an *incentive payment* for cloudlets to reduce the service delay and thus attract more users to win the market.

To make the presentation concise, we use VMs to represent consumers thus buyers while cloudlets to represent the cloud provider thus seller directly in many places.

In the proposed VM-cloudlet matching game, a VM will be matched with a single cloudlet for resources, while one cloudlet can be matched with multiple VMs. A cloudlet can run multiple VMs corresponding to different applications offloaded. We denote the VM set in $cloudlet_j$ as S_j. If VM_i is associated with $cloudlet_j$, we have $VM_i \in S_j$. A cloudlet j and its corresponding VM set S_j form a *coalition* $\widehat{S_j, j}$, whose benefit is the summation of utility from both VM and cloudlet sides:

$$r_{S_j,j} = \sum_{VM_i \in S_j} \pi_i^t \left(j, \vec{p}_{ij}\right) + u_j^{t,load} \left(S_j, \vec{p}_j\right), \tag{1}$$

where $\pi_i^t \left(j, \vec{p}_{ij}\right)$ and $u_j^{t,load} \left(S_j, \vec{p}_j\right)$ are the utilities of VM_i and $cloudlet_j$, which are defined in (8) and (10), respectively. $\vec{p}_{ij} \in R^d$ is the d-dimensional bidding price the user of VM_i offers for using the resources of $cloudlet_j$, and the vector $\vec{p}_j$ includes the bidding prices received by $cloudlet_j$ from the requested users.

The game consists of multiple VM-cloudlet coalitions, and the system's benefit is the summation of benefits of all coalitions.

$$\sum_{1 \le j \le m} r_{S_j,j} \tag{2}$$

Our *goal* is to find the optimal set of coalitions with optimal prices that maximize the system's benefit:

$$\begin{aligned} \max \quad & \sum_{1 \le j \le m} r_{S_j,j} \\ s.t. \quad & \sum_{VM_i \in S_j} \vec{v}_i\left[k\right] \le \vec{s}_j\left[k\right] \\ & 1 \le k \le d. \end{aligned} \tag{3}$$

where $\sum_{VM_i \in S_j} \vec{v}_i\left[k\right] \le \vec{s}_j\left[k\right]$ denotes the capacity constraint of the cloudlet.

Designing such a pricing mechanism is highly nontrivial. It requires a joint solution of resource allocation and on-line offloading scheduling while concurrently considering multiple types of resources. Either of the two problems is a challenging task by itself. In addition, a price charged from the cloudlet or a bidding price offered by the mobile user will impact the willingness to participate in game from the other side. Users and cloudlets need to well negotiate price to achieve efficient coalition.

In order to solve the problem in (3), in following two sections, we first design the utility functions that can better capture the user preference and performance impact, and then develop distributed algorithm that can solve the problem efficiently through price negotiation.

IV. Utility Function to Efficiently Capture the User Preferences and Performance Impacts

The preference of each player in the game is represented by a utility function. To encourage more efficient cloud resource usage while achieving a higher service performance, besides modeling basic utility functions of VMs and Cloudlets, our proposed utility functions take into account the impact of scheduling on service delay and resource usage efficiency.

A. Representing Basic Utility Function

The utility function of a cloudlet is related to its benefit, and depends on VMs it serves and prices it charges the users. The lowest price a $cloudlet_j$ would sell its resources to VM_i is represented as $\vec{\sigma}_{ij}$ to cover its basic cost, where $\vec{\sigma}_{ij} \in R^d$ and d is the total number of resource types in the cloudlet. Denote S_j as the set of VMs selected to run in $cloudlet_j$, the net benefit of $cloudlet_j$ can be expressed as

$$u_j(S_j, \vec{p}_j) = \sum_{VM_i \in S_j} \Big(\sum_{1 \le k \le d} \vec{p}_{ij}[k] \cdot \vec{v}_i[k] - \sum_{1 \le k \le d} \vec{\sigma}_{ij}[k] \cdot \vec{v}_i[k] \Big), \tag{4}$$

where $\sum_{1 \le k \le d} \vec{p}_{ij}\left[k\right] \cdot \vec{v}_i\left[k\right]$ and $\sum_{1 \le k \le d} \vec{\sigma}_{ij}\left[k\right] \cdot \vec{v}_i\left[k\right]$ denote the revenue and cost for $cloudlet_j$ to provide resources to VM_i, respectively. If the prices of cloudlets are very high, users may choose to use the remote cloud for offloading. In order to win the services, a cloudlet can take the cost of the remote cloud as well as its infrastructure cost as guidelines to set its basic prices.

The utility function of a VM is related to its net benefit, which depends on the resources provided and the price charged by a selected cloudlet. The benefit of a VM increases as it obtains more resources, and decreases if the resource price is higher. The benefit of VM_i when associating with $cloudlet_j$ can be

$$\pi_i \left(j, \vec{p}_{ij}\right) = \sum_{1 \le k \le d} w_i \cdot a_{ik} \cdot \vec{v}_i\left[k\right] - \sum_{1 \le k \le d} \vec{p}_{ij}\left[k\right] \cdot \vec{v}_i\left[k\right] \tag{5}$$

where the constant a_{ik} represents the unit gain of resource dimension k of VM_i. We introduce the weight factor w_i to capture the priority of a VM's application so that higher priority application can have better resource access. The payment from VM_i to $cloudlet_j$ for resources is $\sum_{1 \le k \le d} \vec{p}_{ij}\left[k\right] \cdot \vec{v}_i\left[k\right]$.

B. Incorporating the Impact of Service Delay

Generally, the overall service delay includes the response time for an application request, the time consumed for data transmission and the time for executing the tasks on a cloud/cloudlet. The request response time t_i for an application i is the time duration between the arrival of the application request at a cloudlet and time that the requested application is offloaded to the cloudlet for resources.

With only the basic utility, a cloudlet would prefer to serve applications with higher resource requests to make more profit, without motivation to serve users with higher delay. Thus the VM requests with larger resource requirements are usually selected first, resulting in an unfair VM schedule and consequently significant performance degradation for applications with lower resource requirements. Specifically, the offloading sequence directly impacts the application performance. For example, in Fig.3, VM_2 request arrives much earlier and should be executed first although it has lower resource requirement; otherwise, the long waiting will significantly impact its satisfaction.

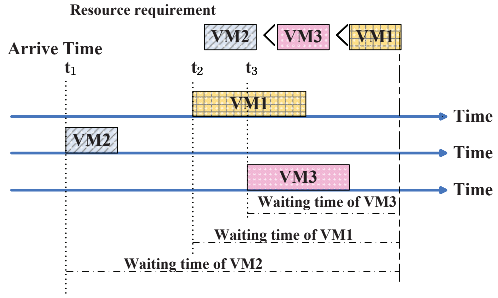


Fig. 3. An example of request arrivals.

If a job requested by a user is delayed in response, the benefit from the job may reduce even if it is executed in the end. The longer the waiting time, the larger the benefit loss. In this paper, we use $g\left(t_i\right) = \varpi_i \cdot w_i \cdot t_i$ to represent the benefit loss of VM_i, where w_i is the weight reflecting the priority of VM_i and ϖ_i is a constant that represents the lost gain per unit time. ϖ_i can be determined by the mobile user i according to its lost perceived quality due to the job delay.

As mentioned in the system model in Section III, to reduce the response time, the platform would provide an incentive payment to encourage a cloudlet to provide resources to delayed VMs. We apply $p(t_i) = \theta_i \cdot w_i \cdot t_i$ to represent the encouragement benefit paid by the platform to the cloudlet to serve VM_i, where θ_i is a constant that represents the price per unit of waiting time.

In turn, to encourage a delayed VM to utilize the resources of a cloudlet, the cloudlet will give the VM some refund, called penalty cost in this paper. We use $f_j(t_i) = \vartheta_{ij} \cdot w_i \cdot t_i$ to represent the penalty cost paid by $cloudlet_j$ to VM_i if there is a service delay, where ϑ_{ij} represents the price per unit of waiting time.

Therefore, we extend the utility functions of cloudlet and VM to incorporate the response time for a VM request:

$$u_j^t(S_j, \vec{p}_j) = \sum_{VM_i \in S_j} \left(\begin{array}{l} \sum\limits_{1\le k\le d} \vec{p}_{ij}[k] \cdot \vec{v}_i[k] - \sum\limits_{1\le k\le d} \vec{\sigma}_{ij}[k] \cdot \vec{v}_i[k] \\ -f_j(t_i) + p(t_i) \end{array} \right) \tag{6}$$

and

$$\pi_i^t(j, \vec{p}_{ij}) = \sum_{1\le k\le d} w_i \cdot a_{ik} \cdot \vec{v}_i[k] - \sum_{1\le k\le d} \vec{p}_{ij}[k] \cdot \vec{v}_i[k] + f_j(t_i) - g(t_i) \tag{7}$$

On the cloudlet side, as cloudlets are selfish and rational, they tend to serve the VMs that can maximize their benefits. To encourage the cloudlets in the game to serve the delayed application requests first, we have $p(t_i) \geq f_j(t_i)$ thus $\theta_i \geq \vartheta_{ij}$. On the VM side, as mobile users are also selfish and rational, a mobile user usually offloads its delayed application to the cloudlet which brings it the maximum benefit. As a cloudlet can make profit only when serving the mobile user, in our system, the cloudlet will set the ϑ_{ij} to a large value to cover the mobile user's cost due to job delay in order to attract more users and win the market. That is $f_j(t_i) \geq g(t_i)$, thus, $\vartheta_{ij} \geq \varpi_i$. Therefore, for the parameter setting, we have $\theta_i \geq \vartheta_{ij} \geq \varpi_i$. In the simulation part, we will show that the control of application waiting time helps to support fair resource allocation and scheduling.

Besides the response time, the transmission delay and execution delay should also be incorporated into the utility function. Let $RTT_{i,j}$ denote the RTT delay between the mobile user of VM_i and the $cloudlet_j$, and e_{ij} denote the delay for $cloudlet_j$ to complete the application i. The RTT delay may be estimated from the transmission time of the request and acknowledgement with the response delay at the cloudlet deducted, and the response message from a cloudlet j can include the estimated application execution time (normalized for comparison across cloudlets), which depends on the load and capacity of the server. A delay-sensitive application, especially the one with the frequent interaction with the cloud, will not select a server far away.

To reflect the whole service delay, we update the utility function in (5) by incorporating these delay cost $f(RTT_{i,j}, e_{ij})$

$$\begin{array}{l} \pi_i^t(j, \vec{p}_{ij}) = \sum\limits_{1\le k\le d} w_i \cdot a_{ik} \cdot \vec{v}_i[k] - \sum\limits_{1\le k\le d} \vec{p}_{ij}[k] \cdot \vec{v}_i[k] + f_j(t_i) - g(t_i) \\ +f(RTT_{i,j}, e_{ij}) \end{array} \tag{8}$$

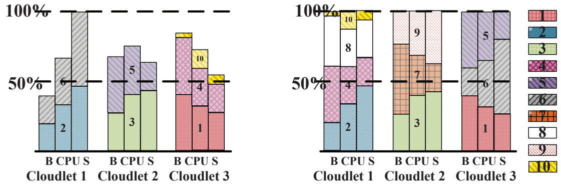
As an example, $f(RTT_{i,j}, e_{ij})$ can be defined as

$$f(RTT_{i,j}, e_{ij}) = \begin{cases} \gamma_i(e^{RTT_{i,j}+e_{ij}}) & \text{delay sensitive application} \\ \gamma_i(RTT_{i,j}+e_{ij}) & \text{otherwise} \end{cases} \tag{9}$$

where γ_i is a constant weight.

C. Enabling Efficient Offloading Scheduling in the Presence of Dynamic Request Arrivals

The application offloading sequence directly impacts the resource utilization thus the benefits of both sides, cloudlets and users. Fig.4 has 10 VM requests and three cloudlets. VM_1, VM_2, and VM_3 arrive first, which are followed by VM_4, VM_5, and VM_6, and then VM_7, VM_8, VM_9 and VM_{10}. Each cloudlet has limited three dimensional resources, including Bandwidth (B), CPU and Storage (S). If a cloudlet only seeks to obtain high profit in the short term, it would select VMs with the total largest resource requirement, which may result in an inefficient resource allocation as shown in Fig.4(a). Although the storage, CPU, and bandwidth utilizations in cloudlets 1, 2 and 3 are high, the bandwidth utilization in cloudlet 1, and storage utilizations in cloudlets 2 and 3 are only 40%, 70% and 60%, respectively.



(a) An inefficient allocation

(b) An efficient allocation

Fig. 4. An example of multi-dimensional resource balancing.

Ideal scheduling may be performed if the resource requests of all applications are known in advance. However, in a practical system, applications arrive dynamically. The greedy service scheduling would lead to extremely unbalanced resource utilization in different dimensions (Fig.4(a)), thus only 7 VMs can be served while leaving the other 3 waiting for resources. As an alternative, with balanced resource utilization in Fig.4(b), the utilizations of all three resources are nearly 100% and all the 10 VMs are served. Therefore, without knowing application offloading requests in advance, balancing resource utilization in multi-dimensions may be an option for cloudlets to accommodate more VMs. The increase of resource usage efficiency not only allows cloudlets to make more profits, but also allows more VMs to run concurrently, which helps reduce the response time of applications and energy consumption of mobile devices.

Therefore, balancing the resource allocation across different dimensions helps cloudlets to obtain higher benefit in the long run. To take into account the impact of multi-dimensional load for more balanced resource provisioning, we introduce a benefit discount factor $f_{S_j,j}^{index}$ into the utility function of cloudlet:

$$u_j^{t,load}(S_j, \vec{p}_j) = u_j^t(S_j, \vec{p}_j) \cdot f_{S_j,j}^{index}. \tag{10}$$

where $f_{S_j,j}^{index}$ is the benefit discount factor designed based on Jain's fairness index [47]. Jain's fairness index is a widely utilized metric to evaluate the fairness resource allocation in network field [48]. $f_{S_j,j}^{index}$ is calculated as

$$f_{S_j,j}^{index} = \frac{\left(\sum\limits_{1\le k\le d} \left(\sum\limits_{VM_i \in S_j} v_i[k] \right) \right)^2}{d \cdot \sum\limits_{1\le k\le d} \left(\sum\limits_{VM_i \in S_j} v_i[k] \right)^2} \tag{11}$$

Obviously, $f_{S_j,j}^{index}$ is a function of all the offloaded VM resource requirements in the $cloudlet_j$. A higher value $f_{S_j,j}^{index}$ indicates the resource consumptions among multiple dimensions of all the VMs offloaded on the cloudlet are more balanced distributed. The $f_{S_j,j}^{index}$ is 1 if all resource dimension has equal load, and is $1/d$ if all resource usages are associated with only one dimension and all other dimensions are idle. Obviously, $f_{S_j,j}^{index}$ is bounded between $1/d$ and 1. According to the bound of the the benefit discount factor, we have

$$u_j^t(S_j, \vec{p}_j) \cdot \tfrac{1}{d} \leq u_j^{t,load}(S_j, \vec{p}_j) \leq u_j^t(S_j, \vec{p}_j) \cdot 1 \tag{12}$$

$u_j^{t,load}(S_j, \vec{p}_j) = u_j^t(S_j, \vec{p}_j)$ if a VM-cloudlet matching makes balanced utilization of all d-dimensional resources in $cloudlet_j$, that is, $f_{S_j,j}^{index} = 1$.

V. Distributed Algorithm for Efficient Resource Negotiation

In this section, we first present the proposed price-adjustment algorithm, and then provide our solution to fairly allocate cloud resources to mobile users in a dynamic mobile environment.

A. Price-Adjustment Algorithm

For the incentive framework to function, there needs an algorithm to facilitate price negotiation between cloudlets and users for efficient resource provisioning and offloading scheduling. Different from a traditional auction-based pricing mechanism which requires a central auctioneer, we develop a distributed algorithm to facilitate the price negotiation between cloudlets and users and match user VM requests with nearby cloudlets.

The algorithm is executed iteratively. In each iteration, there are two phases, including cloudlet selection and user request update as shown in Algorithm 1. When a user wants to offload its application to mobile cloud, the user first broadcasts its VM request for an application, nearby cloudlets will send back their basic charges based on their system cost and also the current average execution time of a task measurement unit. Cloudlet cannot accept a price smaller than its basic cost. The initial bidding price each VM_i offers to one $cloudlet_j$ could be set to $\vec{p}_{ij} = \vec{\sigma}_{ij}$. After a user ascertains the candidate cloudlets, the user will send an offloading request to its selected cloudlet. A cloudlet that receives the VM requests from the customers can reject all but its preferred VMs based on the bidding prices. A mobile user is allowed to offload its application to a cloudlet only when its corresponding VM is selected by the cloudlet. The requests rejected in one iteration by one cloudlet may be repeated in the next iteration where the user can increase its VM bidding price for this cloudlet by a unit price vector $\vec{\delta}$.

In step 9 of Algorithm 1, the number of favorite VMs admitted by a $cloudlet_j$ is $n_j\left(\vec{p}_{j(t)}\right) = \left\lfloor \zeta\left(\vec{p}_{j(t)} - \vec{\sigma}_j\right) + 1 \right\rfloor$, where $\vec{p}_{j(t)}$ is the average bidding price received by the $cloudlet_j$ at the iterative step t (i.e.,$\vec{p}_{j(t)}$), $\vec{\sigma}_j$ is the basic price of $cloudlet_j$, and ζ is a constant. The value 1 in $n_j\left(\vec{p}_{j(t)}\right)$ is added to guarantee that at least one VM is admitted to use the cloud resource in an iteration.

To reduce the payment, when being rejected by a cloudlet, a user has the option of increasing its price bid for one type of resource corresponding to one price dimension at a time. That is, $\vec{\delta}$ is in the set $\{\{0,0,\delta\},\{\delta,0,0\},\{0,\delta,0\}\}$, where δ is the unit price-step. As a simple strategy, an ***un-bias price strategy*** can be taken, with the price vector for a given cloudlet increasing with the unit price vector $\vec{\delta}$ following the sequence, $\{\{0,0,\delta\},\{\delta,0,0\},\{0,\delta,0\}\},\{\{0,0,\delta\},\{\delta,0,0\},\{0,\delta,0\}\},\cdots$. Alternatively, users could increase price for multiple dimensions at the same time to reduce the time of negotiation at higher payment.

Essentially, prices are negotiated among users and cloudlets. As cloudlets need little information from the users, its naturally reduces the complexity of the algorithm and the overhead for implementation.

B. Handling Dynamic Offloading Request and User Mobility

In mobile wireless networks, applications may generate offloading requests dynamically at uncertain time. As resources in cloudlets are limited, to prevent a cloudlet from being occupied by some users for a very long period of time, we divide the service time into reasonable size of slots taking into account the overhead of VM uploading and service fairness.

Algorithm 1 The Distributed Price-Adjustment Algorithm

```
 1: **Initialization** (For every user VM request)
 2: The user broadcasts its VM request, nearby cloudlets will send back their
    basic charges, then the user sets its initial bidding prices $\vec{p}_{ij} = \vec{\sigma}_{ij}$ for all
    nearby cloudlets.
 3: Compute its waiting time according to its arriving time.
 4: Calculate the utility values in Eq.(8) based on the bidding prices and its
    waiting time.
 5: Identify the cloudlet that makes the VM's utility value larger than 0 as the
    VM's available cloudlet.
 6: Select its favorite cloudlet from the set of available cloudlets according to
    the utility values, and send its offloading request to this cloudlet.
 7: **while** user's offloading request or cloudlet's reject is issued, **do**
 8:    **Cloudlet side: Cloudlet Selection** (For cloudlet which receives the
       offloading requests from users)
 9:    Select its favorite VMs according to its utility value calculated by Eq.(10)
       and under the constraint of its remaining resources.
10:    Reject all but its selected favorite VMs.
11:    **User side: User Requesting Update** (For user whose offloading request
       is rejected by one cloudlet)
12:    **if** $VM_i$'s request is rejected by $cloudlet_j$, **then**
13:       $VM_i$ must raise the bidding price by $\vec{\delta}$ to $cloudlet_j$, that is $\vec{p}_{ij} =$
          $\vec{p}_{ij} + \vec{\delta}$.
14:    **else**
15:       For other $cloudlet_{j'}$, $VM_i$'s bidding price remain the same, that is
          $\vec{p}_{ij'} = \vec{p}_{ij'}$.
16:    **end if**
17:    Calculate the utility values in Eq.(8) and identify the set of its available
       cloudlets.
18:    **if** no available cloudlet exists, **then**
19:       No offloading request is sent out.
20:    **else**
21:       Select its favorite cloudlet from the set according to the utility values,
          and send offloading request with new bidding price to the selected
          cloudlet.
22:    **end if**
23: **end while**
```

At the beginning of a time slot, the distributed price-adjustment algorithm is executed for users and cloudlets to negotiate resource allocations and settle the prices. If a VM of an application is scheduled to execute in a time slot, the application arriving time is updated to the end of the time slot, and the VM will compete with others to utilize the resources in the next time slot; otherwise, the application arriving time will not change and the increase of the waiting time will allow an application a higher chance of being scheduled in future time slots with the incentive payment to the cloudlets in Eq.(6). Our performance studies indicate that our design can constrain the maximum response time even when the application arriving rate is large, which demonstrates the effectiveness of our fair scheduling design.

After offloading his applications to a cloudlet associated with an AP nearby, a user may move out of the communication range of its current AP. The cloudlet can transmit the results to the mobile user to a new AP through Inter-AP communications. A user with additional application tasks can also negotiate resources for the new tasks with the available cloudlets at the new location. The user mobility also contributes to the unpredictable VM arrivals to a cloudlet, and our algorithm is designed to encourage balanced resource provisioning to accommodate more VMs in the presence of dynamic VM requests.

VI. Game Analysis

In this section, we first analyze the properties of the proposed game and utility, then prove that the proposed price-adjustment algorithm makes the VM-cloudlet matching game converge to an equilibrium at which the matching between VMs and cloudlets is stable. Finally, we prove that this equilibrium is a *core allocation* at which the mobile cloud system obtains the maximum system benefit.

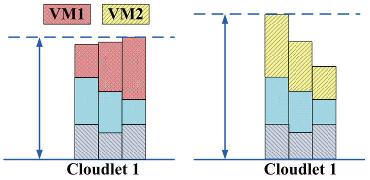
1) Service Delay Effect to the Game

In the extended utility function of Eq.(6), we have added a penalty payment from the cloudlet to the delayed VM and an encouragement payment from the platform to the cloudlet to provide an incentive for the cloudlet to schedule a VM with a longer waiting time earlier. In the example shown in Fig. 3, the cloudlet will have an incentive to schedule VM_2 to execute as soon as possible as otherwise it needs to pay a high penalty charge to VM_2. As a result, our game can provide an efficient VM execution schedule to reduce the response time to VMs.

Besides considering the request response time, we also incorporate the transmission delay and execution delay into the extended user utility function Eq.(8).

2) Multi-dimensional Load Effect to the Game

To obtain higher resource utilization in cloudlets, we have added a load aware benefit discount factor in the utility function of cloudlet in Equ.(10) as the incentive for the cloudlet to select VMs that can make resource utilization among different dimensions more evenly distributed.



(a) VM_1 to $cloudlet_1$ (b) VM_2 to $cloudlet_1$

Fig. 5. Multi-dimensional load effect to the game.

Given that two VM_1 and VM_2 compete in using the resources of $cloudlet_1$ as in Fig.5, if the cloudlet could make the same profit by serving either of them, the cloudlet will prefer to choose VM_1 to use its resources in a balanced way. As a result, with a well-designed benefit discount factor in (10), our game encourages the cloudlet to balance its load across multiple dimensions, which in turn allows the cloudlet to accommodate more VMs with dynamic and unpredicted application requests in the longer term.

A. The Core of Game

Our VM-cloudlet matching game is designed based on the two-sided market game. The two-sided market game is one of the coalitional games that usually use the term core instead of equilibrium to denote the steady state when no buyer or seller is able to negotiate an agreement that is mutually more beneficial compared with that of the current. Therefore, we use the term *core* instead of *equilibrium* in this paper. To analyze the game, we first give the definition on the core of our game.

We use $\mu : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, m\}$ to denote the assignment function, where $\mu(i)$ represents the cloudlet to which the VM_i is assigned and $S_j = \{i\,|j = \mu(i)\}$ is the set of VMs which use the resources in $cloudlet_j$.

Definition 1: An individual rational allocation $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$ is defined as a matching μ between VMs and cloudlets with a price vector $\vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}$ such that,

$$\vec{p}_{i\mu(i)} \geq \vec{\sigma}_{i\mu(i)} \qquad (13)$$

$$\pi_i^t(j, \vec{p}_{ij}) \geq 0 \qquad (14)$$

Eq.(13) means that a seller (cloudlet) will only accept the request with the bidding price larger than its basic price, so its benefit can be positive. Eq.(14) means a buyer (VM) will not buy resources with prices that make its benefit negative.

Definition 2: A discrete core allocation is an individual rational allocation $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$, such that there is no VM-cloudlet coalition $\widehat{S,j}$ with price vector $\vec{r}_j = \vec{r}_{1j}, \cdots, \vec{r}_{ij}, \ldots, \vec{r}_{nj}$ that satisfies:

$$\pi_i^t(j, \vec{r}_{ij}) \geq \pi_i^t\left(\mu(i), \vec{p}_{i\mu(i)}\right) \text{for all } i \in S \qquad (15)$$

$$u_j^{t,load}(S, \vec{r}_j) \geq u_j^{t,load}\left(S_j^\mu, \vec{p}_j\right) \qquad (16)$$

where S_j^μ denotes the set of VMs assigned to $cloudlet_j$ by μ. The first inequality in (15) says that every VM_i in the set S prefers to buy resource from $cloudlet_j$ at price $\vec{r}_{ij}$ rather than buy resource from $cloudlet_{\mu(i)}$ at $\vec{p}_{i\mu(i)}$. The second inequality in (16) says that $cloudlet_j$ can make a higher profit by selling its resource to VMs in S with price $\vec{r}_j = \vec{r}_{1j}, \cdots, \vec{r}_{ij}, \ldots, \vec{r}_{nj}$, than by selling resources to VMs in S_j^μ assigned by μ at the current price $\vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}$. The $\vec{p}_j$ in Eq.(16) is the current bidding prices received by $cloudlet_j$. If these two inequalities are satisfied for a coalition $\widehat{S,j}$ at price $\vec{r}_j$, then this coalition is said to be capable of improving upon its current allocation with the assignment according to μ.

We assume the price in this paper is a discrete value, and the core in our VM-cloudlet matching game is called the discrete core. From the Definition 2, obviously, we can conclude that the discrete core allocation is stable and not any VM-cloudlet coalition can improve upon the allocation. In the following section, we will show an important property of our VM-cloudlet game that the core allocation in the game achieves the Pareto efficiency in terms of the total profit of the system (i.e., bringing together the utility values of all players).

B. Properties of the Algorithm

Theorem 1. *The algorithm will converge in a finite number of iterations to a discrete core allocation.*

Before we prove the Theorem 1, we give the following two lemmas.

Lemma 1. *After a finite number of iterations, no requests are sent by VMs till the algorithm stops.*

Proof: In our algorithm, a cloudlet in one iteration can receive multiple VMs' requests and only admit some of them. If a VM request is rejected, it will increase its price. As the price is a certain discrete value, after a finite number of iterations, a VM will not increase its bidding price for a cloudlet any more as an extremely high price would make its utility corresponding to the cloudlet to be negative, and consequently the VM will stop sending the request to the cloudlet. After a period of time, no VMs will issue the requests to any cloudlets regardless that a VM is admitted or rejected. ■

Lemma 2. *The algorithm converges to an individually rational allocation.*

Proof: Let l^* be the step at which the process stops, and let μ denote the assignment to which it converges. Then $\vec{p}_{i\mu(i)}(l^*) \geq \vec{\sigma}_{i\mu(i)}$ is held for all cloudlets at step l^*, as a cloudlet

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2642182, IEEE Transactions on Services Computing

will never admit VMs that cannot cover its basic cost. The VM can't indefinitely raise the price and gain a negative utility value in (8), then we get $\pi_i^t\left(\mu\left(i\right), \vec{p}_{ij}\left(l^*\right)\right) \geq 0$ immediately. Therefore, both the inequities in (13) and (14) are satisfied, which completes the proof. ■

Based on above two lemmas, we prove the Theorem 1.

Proof: By Lemma 1 and Lemma 2, the algorithm stops in a finite number of steps to an individually rational allocation denoted as $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$, where $\mu : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, m\}$ is the matching function defined above. Let S_j^μ denote the set of VMs assigned to $cloudlet_j$ by μ. Suppose, by way of contradiction, that $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$ is not a discrete core allocation, there must exist a VM-cloudlet coalition $\widehat{S,j}$ with price vector $\vec{r}_j = \vec{r}_{1j}, \cdots, \vec{r}_{ij}, \ldots, \vec{r}_{nj}$ that satisfies:

$$\pi_i^t\left(j, \vec{r}_{ij}\right) \geq \pi_i^t\left(\mu\left(i\right), \vec{p}_{i\mu(i)}\right) \text{for all } i \in S \qquad (17)$$

$$u_j^{t,load}\left(S, \vec{r}_j\right) \geq u_j^{t,load}\left(S_j^\mu, \vec{p}_j\right) \qquad (18)$$

Because the algorithm converges to $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$ and the cloudlet only admits its favorite VMs in the algorithm, by (18), we can conclude that $cloudlet_j$ must never have received (and therefore never have rejected) the request from VM_i at $\vec{r}_{ij}$ or greater. Since permitted prices never fall, then $\vec{p}_{i\mu(i)} \leq \vec{r}_{ij}$ for all VM_i in S. This contradicts (17), which completes the proof. ■

In Theorem 1, we have proven that our price-adjustment algorithm converges in a finite number of iterations. In each iteration step, a cloudlet can receive multiple VM requests and only $n_j\left(\vec{p}_{j(t)}\right)$ favorable VMs are admitted to use the resource, where $\vec{p}_{j(t)}$ is the average bidding price received by the $cloudlet_j$ at the time step t (i.e., $\vec{p}_{j(t)}$). Therefore, in each step, total $f_t = \sum_{j=1}^{m} n_j\left(\vec{p}_j\left(t\right)\right)$ VMs are admitted to use the resource in the system, where m is the number of cloudlets in the system. When the total amount of resource is enough for the users, all VMs will be admitted to use the cloud resource and we have $\sum_{t=1}^{T} f_t = n$, where n is total number of VMs and T is the total iteration steps needed. Obviously, as $n_j\left(\vec{p}_{j(t)}\right) = \left\lfloor \zeta\left(\vec{p}_{j(t)} - \vec{\sigma}_j\right) + 1 \right\rfloor$, f_t is an approximately linear increasing function of $\vec{p}_j\left(t\right)$ and thus an approximately linear increasing function of the price step δ. Therefore, the convergence complexity can be approximately calculated as $O\left(\sqrt{\frac{n}{\delta m}}\right)$. In the simulation part, we will further show the simulation result on the convergence speed impacted by price step (δ), the number of VMs (n), and the number of cloudlets (m).

C. Properties of the Core

Theorem 2. *With sufficiently small unit of price, the maximum system benefit in (3) is achieved in the core allocation defined in Definition 2.*

Proof: By way of contradiction, suppose that our VM-cloudlet matching game has no strict core allocation to achieve the Pareto efficiency of the system. Assume that there is an individually rational allocation $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$, and let S_j^μ denote the set of VMs assigned to the $cloudlet_j$ by μ. Let $\vec{\rho}_{ij}$ be defined by the equation $\pi_{i,j}^t\left(j, \vec{\rho}_{ij}\right) = \pi_{i,\mu(i)}^t\left(\mu(i), \vec{p}_{i\mu(i)}\right)$. We can define the total gain realized by the coalition of $\widehat{S,j}$ as follows

$$D\left[\left(S, j\right); \mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right] = r_{S,j} - r_{S_j^\mu,j} \qquad (19)$$

where $r_{S,j} = \sum_{\vec{v}_i \in S} \pi_i^t\left(j, \vec{\rho}_{ij}\right) + u_j^{t,load}\left(S, \vec{\rho}_j\right)$ and $r_{S_j^\mu,j} = \sum_{\vec{v}_i \in S_j^\mu} \pi_i^t\left(j, \vec{p}_{ij}\right) + u_j^{t,load}\left(S_j^\mu, \vec{p}_j\right)$ are the total utility value of $\widehat{S,j}$ and $\widehat{S_j^\mu,j}$ by using (1). Because $\pi_{i,j}^t\left(j, \vec{\rho}_{ij}\right) = \pi_{i,\mu(i)}^t\left(\mu(i), \vec{p}_{i\mu(i)}\right)$, then above Eq.(19) can be written as

$$D\left(\cdot\right) = u_j^{t,load}\left(S, \vec{\rho}_j\right) - u_j^{t,load}\left(S_j^\mu, \vec{p}_j\right) \qquad (20)$$

Define

$$\begin{aligned} &F\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right) \\ &= \max_{(S,j)} D\left[\left(S, j\right); \mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right] \end{aligned} \qquad (21)$$

By hypothesis that the allocation does not achieve the maximum system benefit, $F\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right) > 0$ as long as $\left(\mu; \vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$ is individually rational allocation, since otherwise no VM-cloudlet coalition could improve upon that allocation on the market. We shall now show that $F(.)$ is bounded above zero for all individually rational allocations.

Note that $F(.)$ is continuous in $\vec{p} = \left(\vec{p}_{1\mu(1)}, \cdots, \vec{p}_{i\mu(i)}, \ldots, \vec{p}_{n\mu(n)}\right)$ for any given assignment μ because the maximum of continuous functions is continuous, and let

$$\begin{aligned} G\left(\mu\right) &= \min_{\vec{p}} F\left(\mu; \vec{p}\right) \\ s.t. \quad \pi_{i,\mu(i)}^t &\geq 0 \quad for \quad all \quad i, \quad and \\ u_{S_j^\mu,j}^{t,load} &\geq 0 \quad for \quad all \quad j \end{aligned} \qquad (22)$$

Finally, define

$$H = \min_{\mu \in \Psi} G\left(\mu\right) \qquad (23)$$

where Ψ is the set of all assignment functions $\mu : \{1, 2, \cdots, n\} \rightarrow \{1, 2, \cdots, m\}$. G is well-defined because for any given $\mu \in \Psi$, F is continuous and the feasible region of the problem on the right-hand side of Eq.(22) is nonempty and compact. Further, $G\left(\mu\right) > 0$ for all $\mu \in \Psi$. As noted above, $F\left(\cdot\right) > 0$ everywhere in the feasible region for $\mu \in \Psi$. Finally, H is well-defined and strictly positive because Ψ is a finite set. Thus choosing the unit of price smaller than $\frac{H}{(n+1)\cdot\tau}$ (τ is maximum resource requirement of VM) insures that any individual allocation can be improved by at least one VM-cloudlet coalition.

As proven in Theorem 1, our VM-Cloudlet matching game converges to a discrete core allocation where no cloudlet or VM is able to negotiate an agreement that is mutually more beneficial compared with that of the current. However, the above contents show that under sufficiently small choices of the unit of price, our VM-cloudlet matching game has no core allocation, which contradicts Theorem 1, and the proof is completed. ■

From Theorem 2, we can conclude that although our proposed price-adjustment algorithm is distributed, it can facilitate the system to achieve the core allocation to maximize the system benefit.

VII. Simulation Results and Analysis

In this section, we first evaluate the convergence and competition properties of the proposed mechanism, and then evaluate its performance.

A. Simulation setting

To evaluate the performance of our proposed pricing mechanism, we build a trace-driven simulator where the application resource requirement (task ID, resource request for CPU, resource request for RAM, and resource request for disk space) is extracted from the Google cluster-usage trace [49], [50]. The resource requirements are normalized within [1, 600], [1, 600], and [1, 600] respectively. Accordingly, in the simulated mobile cloud system,

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2642182, IEEE Transactions on Services Computing

each cloudlet has 3-dimensional resources, including CPU, RAM, and disk space. Each cloudlet has the same total normalized capacity of three-dimensional resources (10000, 10000, 10000).

To evaluate how dynamic application tasks impact the performance, the application requests arrive dynamically following a Poisson process with the average rate λ. The task execution duration is randomly generated in [1, 5] time slots. The parameters w_i, a_{ik}, ϖ_i, θ_i, ϑ_{ij}, γ_i, and ζ in the VM utility function and cloudlet utility function are randomly generated in [1, 10], [0.1, 0.5], [0.01,0.05], [0.01,0.05], [0.01,0.05], [0.0001,0.01], and [500,1000], respectively with $\theta_i \geq \vartheta_{ij} \geq \varpi_i$. The three-dimensional basic prices of different cloudlets are randomly generated in [0.05, 0.1].

We implement four pricing mechanisms to compare the performance. The first two pricing mechanisms are based on the proposed pricing-adjustment algorithm, in which the VM chooses the cloudlet that provides it with the maximum utility to send the offloading request, and raises the price when it is rejected by this cloudlet. Then, according to whether penalty payment and benefit discount are adopted in the pricing-adjustment procedure, the pricing mechanisms can be further divided into two different types, denoted as MPTMG-PB and MPTMG-NPNB. In the MPTMG-PB mechanism, VM obtains a penalty payment from a committed cloudlet when its request is not handled in time, and the cloudlet calculates its utility according to (10) by considering the benefit discount due to unbalanced resource utilization. We set $\delta = 0.001$ as the price-step to increase price when a VM is rejected by the cloudlet it sends request to.

The prices in the last two mechanisms are fixed. A VM pays a selected cloudlet at a fixed multi-dimensional price for using resources, and applications are scheduled to offload to cloudlets according to their arriving time. For the VMs with the same arriving time, a VM placement algorithm based on a bin-packing [51] is adopted to place VMs to cloudlets to maximize the resource utilization. In the third pricing mechanism, there are limited 8 types of VMs. Every application will match itself to a type of VM, and a cloudlet allocates resources to the VM according to its type, denoted as LIMITED. While in the fourth mechanism, the VM's resource requirement is heterogeneous similar to the practical application resource requirements, denoted as UNLIMITED. It is worthy pointing out that all our implemented pricing mechanisms exploit multi-dimensional price for resource management.

The metrics used to evaluate different pricing mechanisms are explained as follows: 1) Resource utilization among all cloudlets' resource dimensions; 2) Response time: the time duration that an application has to wait before being offloaded to a cloudlet to run in a VM; 3) System benefit: defined in Eq.(2) as the sum of benefits of all players in the mobile cloud system; 4) VM benefit: defined as the benefit calculated by the utility function in Eq.(8); 5) Cloudlet benefit: defined as the profit calculated by the utility function in Eq.(10); 6) Execution time: the time duration to complete all tasks of application requirements.

B. Simulation results

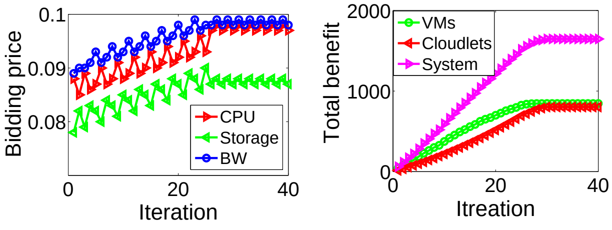
As mentioned in Section V, the proposed distributed price-adjustment algorithm is executed at the beginning of each time unit. In the VM-cloudlet matching game, there are competitions among the players in both sides. To clearly investigate the convergence and market behavior in our proposed pricing mechanism, we firstly run our MPTMG-PB for one time slot.

1) Convergence behavior

We set the number of VMs and cloudlets to 100 and 3 respectively. Fig.6(a) plots the adjustment procedure for the multi-dimensional price of one VM during the iteration, which is shown to approach stable values as the iteration number increases. Initially, each mobile user sets its price as low as possible for different cloudlets. A user increases its bidding price for one cloudlet after the user is rejected by this cloudlet. From Fig.6(a), we find that the bidding price may reduce sometimes when the VM selects an alternate cloudlet to send its request during the iteration procedure.

A user sets its initial bidding prices to cover the basic cost of the resources, that is, $\vec{p}_{ij} = \vec{\sigma}_{ij}$. As different types of resource have different basic cost, the starting point (which corresponds to the value of the basic cost) in the curves of CPU, Storage, BW in Fig.6(a) are different. The price changes in different resources are not synchronized with the iterative process as we adopt the un-bias price strategy (in Section V-A). Although not synchronized, all the price dimensions converge quickly which ensures the cloudlets to quickly provide the VMs for services.

From 6(b), we also observe that the total system utility increases with iterations, and converges to a stable value quickly, which demonstrates that the proposed MPTMG-PB is efficient for on-line resource management.



(a) Bidding price (b) Total benefit

Fig. 6. Convergence behavior.

To investigate how competitions among buyer-side (VMs) impact the convergence behavior, we vary the number of VMs and the number cloudlets, as shown in Fig.7. The total number of iterations increases when the number of VMs becomes larger until the number of VMs reaches a certain value, beyond which the number of iterations becomes stable. Initially, the increase of VMs creates higher buyer competition. As many VMs are rejected by the cloudlets in one iteration, it results in a decrease of the convergence speed. Take the number of cloudlets equal to 3 as an example, because the total resources in the cloudlets of the whole system are limited, the number of iterations maintains the stability after the number of VMs reaches a certain level. When the number of cloudlets increases, the system can admit more VMs in one iteration, therefore the converging speed increases.

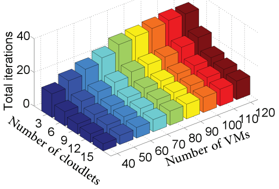


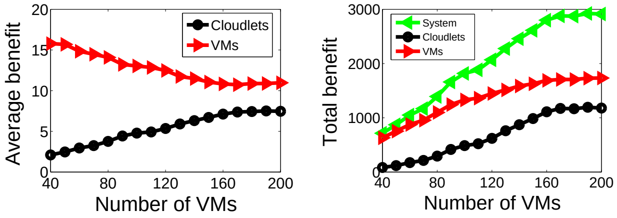
Fig. 7. Impact of competitions among buyer/seller on the convergence behavior.

This article has been accepted for publication in a future issue of this journal, but has not been fully edited. Content may change prior to final publication. Citation information: DOI 10.1109/TSC.2016.2642182, IEEE Transactions on Services Computing

2) Market behavior

We first investigate two factors from both buyer-side (VMs) and seller-side (cloudlets) to analyze how competitions impact the market behavior under our pricing mechanisms. Then we investigate how the price-step $\vec{\delta}$ impacts the market behavior.

- Impact of buyer competition



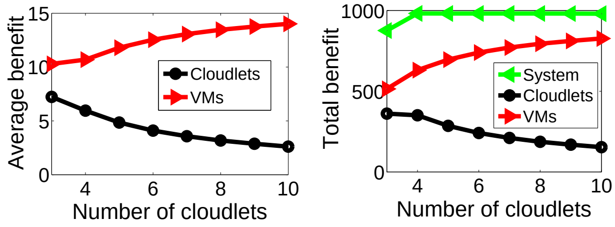
(a) Average benefit (b) Total benefit

Fig. 8. The impact of buyer competition on market behavior.

We vary the number of VMs while keeping the number of cloudlets at 10 in the system. In Fig.8(a), with the increase of the number of VMs thus the increased buyer competition, the average benefit of the admitted VMs decreases. On the other hand, the average profit of cloudlets increases, as shown in Fig.8(a). In Fig.8(b), the total benefit of the mobile cloud system increases when there are more buyers(VMs), and becomes saturated after the number of VMs reaches a large value 180. With the total resources in cloudlets (sellers) limited, the number of VMs can be supported by cloudlets are also restricted.

When there are more VM requests, from the perspective of economic market, the service providers will be attracted to deploy more cloudlets (sellers) for profit.

- Impact of seller competition



(a) Average benefit (b) Total benefit

Fig. 9. The impact of seller competition on market behavior.

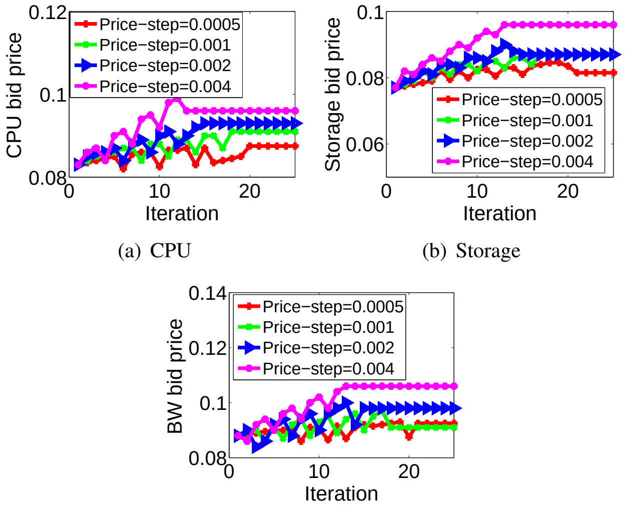
We vary the number of cloudlets while setting the number of VMs to 150. When the number of cloudlets increases thus leading to higher competition among sellers, VMs can buy and use the resources in the cloudlets at lower prices, thus the average benefit of VMs increases. On the other hand, the average profit of cloudlets deceases as shown in Fig.9(a).

As shown in Fig.9(b), due to seller competition, the total benefit of the buyers(VMs) increases while the total benefit of the cloudlets decreases. Moreover, the total benefit of the system increases initially when the number of cloudlets increases, and then remains the same as shown in Fig.9(b). This indicates that the initial number of cloudlets thus resources can not satisfy the total requirements from VMs. When the number of cloudlets increases from 3 to 4, the total amount of resources in the system increases and all these 150 VMs can be admitted to the system, which leads to higher total system benefit. When the number of cloudlets increases beyond 4, the cloudlet resources are more than those required by all the VMs, thus the total system benefit remains same.

As a result of competition among cloudlets, more VMs will be attracted to enter the system and use the resources in the cloudlets.

From above competition analysis, we can conclude that the proposed pricing-mechanisms can provide a healthy competition in the mobile cloud system, which helps to establish a stable mobile cloud market.

- Impact of the price-step δ (impact on user's bid price)



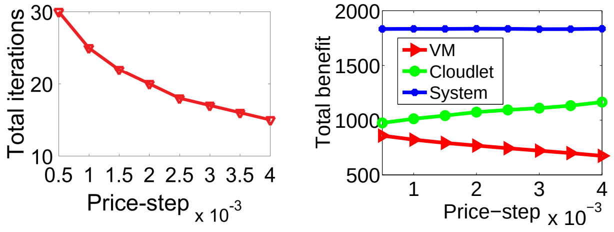
(a) CPU (b) Storage

(c) Bandwidth

Fig. 10. The price-step of δ impacts on the user's bid price.

Fig.10(a), Fig.10(b) and Fig.10(c) depict the bid price of one VM under different δ in different resource dimensions. Similar to Fig. 6, as the different dimension prices are not changed at the same time in a new iteration, there are slight differences in the converging speed of different price dimensions. Obviously, the price-step δ impacts the converging speed. With a larger δ, a VM is easier to be admitted into the system with fewer iterations. The converging speed under $\delta = 0.004$ nearly doubles that under $\delta = 0.0005$.

- Impact of the price-step δ on the system behavior



(a) Bandwidth (b) Total benefit

Fig. 11. The price-step of δ impacts on system's behavior.

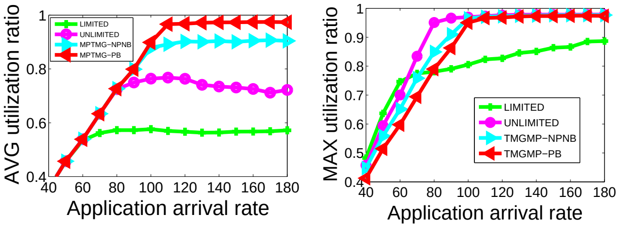
Fig.11(a) and Fig.11(b) depict how δ affects the convergence speed and the total benefit of the system. As expected, the converging speeds increase with the increase of price-step δ. With a larger δ at coarser price adjustment, the benefits of buyers and sellers decrease and increase respectively, because VMs have to pay higher price to purchase the resources. Although the variation of δ may change the benefit allocation between the buyer and seller sides, it does not affect the total system benefit. There is a tradeoff between the speed and the benefit obtained by the buyer. If a user chooses larger price adjustment step, the corresponding VM will have shorter time to negotiate with the cloudlet and obtain the cloud service, at the cost of lower benefit.

Moreover, from Fig.11(a), we can find that the total number of iterations is controlled within 30 iterations under different price-steps in all scenarios, which demonstrates that the computation cost of our algorithm is not high.

3) Performance comparison

The application arrival rate impacts the application workload in the mobile cloud system. To investigate its impact on the pricing mechanism, we vary λ values while keeping the number of cloudlets at 10, and applications are only generated during the first 50 time slots in one simulation run. We analyze the performance results from the perspectives of both the cloud providers and end mobile users.

- Resource utilization



(a) Average resource utilization (b) Maximum resource utilization

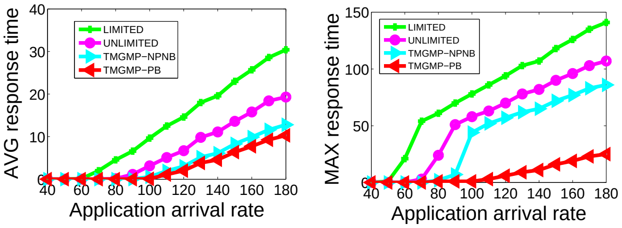
Fig. 12. Resource utilization.

The average resource utilization ratio can evaluate the efficiency of resource allocation, while the maximum resource utilization ratio can reflect the load of the system. If the difference between the above two utilization ratios is large, one resource dimension would become the bottleneck. From Fig.12(a) and Fig.12(b), the average resource utilization ratio and the maximum resource utilization ratio increase as λ becomes larger and then stay stable when λ is beyond a value. After λ is higher than a threshold value, the system becomes congested because of the limited amount of resource in cloudlets.

Compared with other pricing mechanisms, our MPTMG-PB mechanism has the highest average resource utilization ratio that is nearly equal to the maximum resource utilization ratio even when the system becomes congested, which proves that the benefit discount design in MPTMG-PB with the load-balancing resource allocation is effective in improving the resource usage efficiency. In LIMITED, every application matches itself with a type of VM with the resources allocated according to the upper limit of the VM, which is usually larger than the application's resource requirement. As a result, the cloudlet's resource is wasted. Due to the benefit discount design, the average resource utilization in our MPTMG-PB increases by 10%, 30%, and 40% from that under MPTMG-NPNB, UNLIMITED, and LIMITED, respectively.

- Response time

High efficient resource allocation in cloudlets can increase the number of VMs that can run concurrently, and consequently can reduce the waiting time for applications to be scheduled to offload to cloudlets which in turn helps save the energy of mobile users. This impact is demonstrated by examining the average response time, as shown in Fig.13(a). When $\lambda = 180$, the average response time of MPTMG-PB is 20%, 50% and 70% lower than those of MPTMG-NPNB, UNLIMITED, LIMITED. Generally, idle waiting will significantly degrade the user's satisfaction level. Only when the response time is low, a user is willing to enter the system. Compared to other pricing mechanisms, our MPTMG-PB mechanism can help the mobile cloud system to attract more users and win the market.



(a) Average response time (b) Maximum response time

Fig. 13. Response time.

The maximum response time directly reflects the service quality. As shown in Fig.13(b), our MPTMG-PB has the lowest maximum response time. When $\lambda = 180$, the maximum response time of our MPTMG-PB is 70%, 80% and 90% lower than those of MPTMG-NPNB, UNLIMITED, LIMITED. Therefore, application under MPTMG-PB can be offloaded to execute in a cloudlet within a limited small time, and the SLA agreement and QoS requirement of application can be better satisfied. This result demonstrates that the penalty payment designed in this paper can provide an effective incentive for cloudlets to more fairly serve VMs. By reducing the waiting time of individual VMs, a cloudlet can reduce its extra cost paid to the delayed VM and gain encouragement payment from the platform.

- System benefit

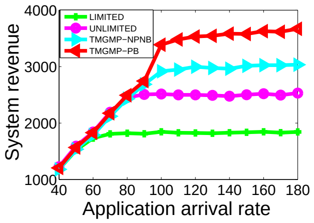


Fig. 14. System benefit.

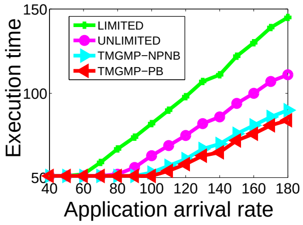


Fig. 15. Execution time.

As proved in Theorem 2, the proposed price-adjustment algorithm allows to obtain the optimal price and VM-cloudlet matching so that the mobile cloud system can achieve Pareto efficiency. As a result, the system benefits of MPTMG-PB and MPTMG-NPNB are significantly higher than the two fixed-pricing mechanisms UNLIMITED and LIMITED, as shown in Fig.14. The total benefit of the system increases when λ increases until it reaches a large value of 110 in MPTMG-PB, beyond which the total benefit is capped due to the limited total cloudlet resources. Moreover, MPTMG-PB outperforms MPTMG-NPNB with up to 20% higher system benefit. As MPTMG-PB can obtain high resource utilization, the cloudlets can make higher profit by selling more resources to VMs while VMs can gain higher benefit by buying resources in cloudlets at lower prices.

- Execution time

Although applications are generated only during the first 50 time slots, as λ increases, the load of the system becomes high. Some applications have to wait for the service, and the time for completing all applications may be larger than 50 time slots. When $\lambda = 180$, MPTMG-PB can finish all tasks using 70 time slots, while MPTMG-NPNB, UNLIMITED, and LIMITED respectively take 80, 100, and 130 time slots to complete all the jobs as shown in Fig.15. The total execution time reduction is important, which helps to reduce power consumption and cost in the mobile cloud system.

In summary, our proposed pricing mechanism MPTMG-PB is designed by considering both resource availability and VM's QoS requirement, so that the optimal system benefit can be achieved while at the same time supporting efficient resource allocation and application offloading.

VIII. Conclusions

The long RTT delay may prevent running delay-sensitive applications in the remote cloud despite its sufficient resources. Although cloudlet can "bring the cloud closer to applications" to mitigate the long latency, compared to a remote cloud, the resources owned by a cloudlet are much more limited. To more efficiently manage application offloading and utilize the limited resources in cloudlets, we design a Multi-dimensional Pricing mechanism based on Two-sided Market Game. Specifically, we propose three types of prices: a multi-dimensional price corresponding to multi-dimensional resource allocation, a penalty price to encourage fair and high quality cloud services, and a benefit discount factor which facilitates more balanced resource allocation to help cloudlets to accommodate more VMs in the long run. We have carried out extensive simulations, and our results demonstrate that both the amount of resource allocation and the sequence of application offloading can be effectively accomplished with the support of our pricing mechanism.

Although this paper focuses on the resource allocation and scheduling in cloudlets, taking into account the impact of transmission delay in the utility function, we can provide a unified service platform consisting of both cloudlets and the remote cloud. A mobile user may choose to use the service provided by the remote cloud if its application is not sensitive to the delay when the cloudlets do not have enough resources to serve all the mobile users.

References

[1] M. Satyanarayanan, R. Schuster, M. Ebling, G. Fettweis, H. Flinck, K. Joshi, and K. Sabnani, "An open ecosystem for mobile-cloud convergence," *IEEE Communications Magazine*, vol. 53, no. 3, pp. 63–70, 2015.

[2] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: architectures, challenges, and applications," *IEEE Wireless communications*, vol. 20, no. 3, pp. 14–22, 2013.

[3] M. Satyanarayanan, Z. Chen, K. Ha, W. Hu, W. Richter, and P. Pillai, "Cloudlets: at the leading edge of mobile-cloud convergence," in *MobiCASE*, IEEE, 2014.

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

[5] M. Satyanarayanan, P. Bahl, R. Caceres, and et.al, "The case for vm-based cloudlets in mobile computing," *IEEE Pervasive Computing.*, vol. 8, no. 4, pp. 14–23, 2009.

[6] K. Ha, P. Pillai, W. Richter, Y. Abe, and M. Satyanarayanan, "Just-in-time provisioning for cyber foraging," in *Mobisys*, 2013.

[7] S. Echeverrıa, J. Root, B. Bradshaw, and G. Lewis, "On-demand vm provisioning for cloudlet-based cyber-foraging in resource-constrained environments," 2014.

[8] G. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Tactical cloudlets: Moving cloud computing to the edge," in *MILCOM*, pp. 1440–1446, IEEE, 2014.

[9] M. Jia, W. Liang, Z. Xu, and M. Huang, "Cloudlet load balancing in wireless metropolitan area networks," in *IEEE INFOCOM 2016-The 35th Annual IEEE International Conference on Computer Communications*, pp. 1–9, IEEE, 2016.

[10] G. A. Lewis, S. Echeverría, S. Simanta, B. Bradshaw, and J. Root, "Cloudlet-based cyber-foraging for mobile systems in resource-constrained edge environments," in *Companion Proceedings of the 36th International Conference on Software Engineering*, pp. 412–415, ACM, 2014.

[11] Q. Xia, W. Liang, and W. Xu, "Throughput maximization for online request admissions in mobile cloudlets," in *Local Computer Networks (LCN), 2013 IEEE 38th Conference on*, pp. 589–596, IEEE, 2013.

[12] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: elastic execution between mobile device and cloud," in *EuroSys*, 2011.

[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "Thinkair: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *INFOCOM*, 2012.

[14] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Mobisys*, 2010.

[15] B.-G. Chun and P. Maniatis, "Dynamically partitioning applications between weak devices and clouds," in *MCS*, 2010.

[16] K. Sinha and M. Kulkarni, "Techniques for fine-grained, multi-site computation offloading," in *CCGrid*, 2011.

[17] Y. Zhang, H. Liu, L. Jiao, and X. Fu, "To offload or not to offload: An efficient code partition algorithm for mobile cloud computing," in *CLOUDNET*, 2012.

[18] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Transactions on Wireless Communications*, vol. 11, no. 6, pp. 1991–1995, 2012.

[19] M. S. D Narayanan, J Flinn, "Using history to improve mobile application adaptation," in *WMCSA*, 2000.

[20] R. W. S Gurun, C Krintz, "Nwslite: a light-weight prediction utility for mobile devices," in *MobiSys*, 2004.

[21] P. W. DT Hoang, D Niyato, "Optimal admission control policy for mobile cloud computing hotspot with cloudlet," in *WCNC*, 2012.

[22] X. Wang and H. Schulzrinne, "Pricing network resources for adaptive applications," *IEEE/ACM Transactions on Networking*, vol. 14, no. 3, pp. 506–519, 2006.

[23] X. Wang and H. Schulzrinne, "Incentive-compatible adaptation of internet real-time multimedia," *IEEE Journal on Selected Areas in Communications*, vol. 23, no. 2, 2005.

[24] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: Towards effective pricing for cloud computing," in *INFOCOM*, 2012.

[25] D. Niu, C. Feng, and B. Li, "A theory of cloud bandwidth pricing for video-on-demand providers," in *INFOCOM*, 2012.

[26] Y. Song, M. Zafer, and K.-W. Lee, "Optimal bidding in spot instance market," in *INFOCOM*, 2012.

[27] B. L. Hong Xu, "Maximizing revenue with dynamic cloud pricing: The infinite horizon case," in *INFOCOM*, 2012.

[28] D. N. Sivadon Chaisiri, Bu-Sung Lee, "Optimization of resource provisioning cost in cloud computing," *IEEE TRANSACTIONS ON SERVICES COMPUTING*, vol. 5, no. 2, pp. 164–177, 2012.

[29] Y. Feng, B. Li, and B. Li, "Bargaining towards maximized resource utilization in video streaming datacenters," in *INFOCOM*, 2012.

[30] B. F. Dorian Minarolli, "Utility-driven allocation of multiple types of resources to virtual machines in clouds," in *2011 IEEE Conference on Commerce and Enterprise Computing*, pp. 137–144, 2011.

[31] D. N. Sivadon Chaisiri, Bu-Sung Lee, "Optimal virtual machine placement across multiple cloud providers," in *IEEE APSCC*, 2009.

[32] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Transactions on Signal and Information Processing over Networks*, vol. 1, pp. 89–103, June 2015.

[33] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Transactions on Networking*, vol. 24, pp. 2795–2808, Oct 2016.

[34] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–14, 2014.

[35] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Computer Communications Workshops (INFOCOM WKSHPS), 2014 IEEE Conference on*, pp. 352–357, IEEE, 2014.

[36] L. Xiang, S. Ye, Y. Feng, B. Li, and B. Li, "Ready, Set, Go: Coalesced offloading from mobile devices to the cloud," in *Proc. of IEEE INFOCOM*, 2014.

[37] S. Deng, L. Huang, J. Taheri, and A. Zomaya, "Computation offloading for service workflow in mobile cloud computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. PP, no. 99, pp. 1–1, 2014.

[38] F. Liu, P. Shu, and J. Lui, "Appatp: An energy conserving adaptive mobile-cloud transmission protocol," *IEEE Transactions on Computers*, vol. PP, no. 99, pp. 1–1, 2015.

[39] A. Ghodsi, M. Zaharia, B. Hindman, A. Konwinski, S. Shenker, and I. Stoica, "Dominant resource fairness: fair allocation of multiple resource types," in *USENIX NSDI*, 2011.

[40] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *Parallel and Distributed Systems, IEEE Transactions on*, vol. 26, no. 4, pp. 974–983, 2015.

[41] D. Gale and L. S. Shapley, "College admissions and the stability of marriage," *The American Mathematical Monthly*, vol. 69, no. 1, pp. 9–15, 1962.

[42] T. Eisenmann, G. Parker, and M. W. Van Alstyne, "Strategies for two-sided markets," *Harvard business review*, vol. 84, no. 10, p. 92, 2006.

[43] L. S. Shapley and M. Shubik, "The assignment game i: The core," *International Journal of game theory*, vol. 1, no. 1, pp. 111–130, 1971.

[44] A. E. Roth, "The evolution of the labor market for medical interns and residents: a case study in game theory," *The Journal of Political Economy*, pp. 991–1016, 1984.

[45] N. Economides and J. Tåg, "Network neutrality on the internet: A two-sided market analysis," *Information Economics and Policy*, vol. 24, no. 2, pp. 91–104, 2012.

[46] S. Lamparter, S. Becher, and J.-G. Fischer, "An agent-based market platform for smart grids," in *Proceedings of the 9th international conference on autonomous agents and multiagent systems: industry track*, pp. 1689–1696, International Foundation for Autonomous Agents and Multiagent Systems, 2010.

[47] R. Jain, D.-M. Chiu, and W. R. Hawe, *A quantitative measure of fairness and discrimination for resource allocation in shared computer system*. Eastern Research Laboratory, Digital Equipment Corporation, 1984.

[48] J. Mo and J. Walrand, "Fair end-to-end window-based congestion control," *IEEE/ACM Transactions on Networking (ToN)*, vol. 8, no. 5, pp. 556–567, 2000.

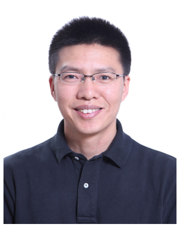
[49] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," *Google Inc., Mountain View, CA, USA, Technical Report*, 2011.

[50] J. Wilkes, "More google cluster data," *Google research blog, Nov*, 2011.

[51] D. J. E Coffman, J Garey, "Approximation algorithms for bin-packing an updated survey," *Algorithm Design for Computing System Design*, 1984.

Kun Xie received PhD degree in computer application from Hunan University, Changsha, China, in 2007. She worked as a postdoctoral fellow in the department of computing in Hong Kong Polytechnic University from 2007.12 to 2010.2. She worked as a visiting researcher in the department of electrical and computer engineering in state university of New York at Stony Brook from 2012.9 to 2013.9. Her research interests include wireless network and mobile computing, network management and control, cloud computing and mobile cloud, and big data. She has published over 60 papers in major journals and conference proceedings (including top journals IEEE/ACM TON, IEEE TMC, IEEE TC, IEEE TWC, and top conferences INFOCOM, ICDCS, SECON, IWQoS)

Xin Wang (M'1 / ACM'4) received the B.S. and M.S. degrees in telecommunications engineering and wireless communications engineering respectively from Beijing University of Posts and Telecommunications, Beijing, China, and the Ph.D. degree in electrical and computer engineering from Columbia University, New York, NY. She is currently an Associate Professor in the Department of Electrical and Computer Engineering of the State University of New York at Stony Brook, Stony Brook, NY. Before joining Stony Brook, she was a Member of Technical Staff in the area of mobile and wireless networking at Bell Labs Research, Lucent Technologies, New Jersey, and an Assistant Professor in the Department of Computer Science and Engineering of the State University of New York at Buffalo, Buffalo, NY. Her research interests include algorithm and protocol design in wireless networks and communications, mobile and distributed computing, as well as networked sensing and detection. She has served in executive committee and technical committee of numerous conferences and funding review panels, and serves as the associate editor of IEEE Transactions on Mobile Computing. Dr. Wang achieved the NSF career award in 2005, and ONR challenge award in 2010.



Gaogang Xie received his B.S. degree in Physics, M.S. degree and Ph.D. degree in computer science all from Hunan University respectively in 1996, 1999 and 2002. He is currently a Professor and Director of Network Technology Research Center with the Institute of Computing Technology (ICT), Chinese Academy of Sciences (CAS), Beijing, China. His research interests include Internet architecture, packet processing and forwarding, and Internet measurement.

Jiannong Cao (M'93-SM'05-FM'14) received the Ph.D degree in computer science from Washington State University, Pullman, WA, USA, in 1990. Dr. Cao is currently a chair professor and head of the Department of Computing at Hong Kong Polytechnic University, Hung Hom, Hong Kong. Dr. Cao's research interests include parallel and distributed computing, computer networks, mobile and pervasive computing, fault tolerance, and middleware. Dr. Cao has served as an associate editor and a member of the editorial boards of many international journals, including IEEE Transactions on Parallel and Distributed Systems, IEEE Network, Elsevier Pervasive and Mobile Computing Journal, Springer Peer-to-Peer Networking and Applications, and Wiley Wireless Communications and Mobile Computing. He has also served as chairs and members of organizing and technical committees of main international conferences, including PERCOM, INFOCOM, ICDCS, DSN, SRDS, ICNP, and RTSS.

Dongliang Xie received the Ph.D. degree from Beijing Institute of Technology, China, in 2002. He is currently a visiting researcher in the department of electrical and computer engineering in state university of New York at Stony Brook. His research interests focus on resource-constrained wireless communication and information-centric network, including architecture of ubiquitous and heterogeneous network, complex network analysis, as well as content retrieval and service management.

Yuqin Ji received M.S. degrees in computer application from Hunan University, China, in 2013. Her research interests include mobile cloud computing.

Jigang Wen received PhD degrees in computer application from Hunan University, China, in 2011. He worked as a research assistant in the department of computing in Hong Kong Polytechnic University from 2008 to 2010. He is now a postdoctoral fellow in Institute of Computing Technology, Chinese Academy of Science, China. His research interests include wireless network and mobile computing, high speed network measurement and management.