The Journal of Systems and Software 152 (2019) 151–164

Contents lists available at ScienceDirect

The Journal of Systems and Software

journal homepage: www.elsevier.com/locate/jss

An auction-based incentive mechanism for heterogeneous mobile clouds

Bowen Zhou[a], Satish Narayana Srirama[b,*], Rajkumar Buyya[a]

[a] Cloud Computing and Distributed Systems (CLOUDS) Laboratory, School of Computing and Information Systems, University of Melbourne, Australia
[b] Mobile & Cloud Lab, Institute of Computer Science, University of Tartu, Estonia

ARTICLE INFO

Article history:
Received 3 November 2018
Revised 5 February 2019
Accepted 6 March 2019
Available online 8 March 2019

Keywords:
Mobile cloud computing
Incentive mechanism
Reverse auction
Code offloading

ABSTRACT

Traditional mobile cloud computing (MCC) adopts a onefold mobile device and public cloud paradigm. As the mobile device capabilities continue to develop, their opportunistic utilization in MCC has recently gained popularity. The new paradigm is a hybrid/heterogeneous mobile cloud (HMC) where mobile devices, cloudlets, and private/public cloud form a shared resource network for task offloading. However, mobile device users are discouraged from sharing their devices for running foreign tasks due to the battery life and privacy concerns. To incentivize mobile device users to utilize and participate in the HMC offloading service, we designed a task offloading market for the HMC service, where a mobile user can compete as a seller with others by bidding its redundant computing resources, and another mobile user as a buyer can pay the bidding price and offload the task to the winning user. To enable an incentive and fair competition of the mobile cloud offloading market, we propose a reverse auction-based incentive mechanism, mCloudAuc, to provide real-time auctions. The proposed auction algorithm demonstrates computation efficiency, truthfulness, and individual rationality for the participants through proof and multiple simulations. Our prototype implementation of mCloudAuc on Android platform has also shown its feasibility in practice.

1. Introduction

The recent innovation and development on mobile devices such as smartphones and tablets have made the smart devices evolve as the primary tool of the digital omnivores in our daily life. According to some recent mobile industry reports (comScore, 2017a, b), mobile device usage has doubled in the past three years and has represented 70% of digital media minutes by the end of 2017. However, the battery life stays the principal weakness of the mobile devices due to the high energy consumption from the display, camera, sensors, etc. that ends up draining the battery quickly, and the battery technology seems unlikely to have a significant improvement in the foreseeable future. Therefore, novel solutions need to be studied to fill the gap, and mobile cloud computing emerges as a promising approach.

Mobile cloud computing is a computing paradigm that enables mobile devices to outsource their computation intensive tasks onto cloud computing resources for execution to conserve the battery on mobile devices and improve the performance of mobile cloud applications. Take optical character recognition (OCR) application for example, a mobile user can take a photo of a piece of paper containing text in a foreign language, and the app will process the photo and show the translated text on the display. Alternatively, the user can subscribe to the cloud service provided by the application provider by paying the subscription fee to offload the photo to cloud services for processing and release the device from halting. However, network bottleneck and lack of performance improvement of offloading to the cloud have been the major problem of adapting the mobile cloud paradigm. For example, a chess game on the Samsung Galaxy S3 will benefit from offloading, only if it is offloaded to a m3.medium or more powerful instance on Amazon EC2, which generates more monetary cost and reduce the user incentive of using mobile cloud offloading services (Srirama, 2017).

More recently, a few research works (Ravi and Peddoju, 2014; Shih et al., 2015; Zhou et al., 2017) have introduced the mobile ad-hoc network (Toh, 2001) and cloudlet (Satyanarayanan et al., 2009) into the original mobile cloud paradigm to form a heterogeneous mobile cloud (HMC), which aims to avert the network bottleneck and ensure adaptation to different offloading conditions. Fig. 1 shows an example of the HMC where the mobile devices in a local area form a wireless ad-hoc network through short-range wireless networks such as WiFi, Bluetooth, and WiFi-direct. In

* Corresponding author.
E-mail addresses: srirama@ut.ee, satish.srirama@ut.ee (S.N. Srirama), rbuyya@unimelb.edu.au (R. Buyya).

https://doi.org/10.1016/j.jss.2019.03.003
0164-1212/© 2019 Published by Elsevier Inc.

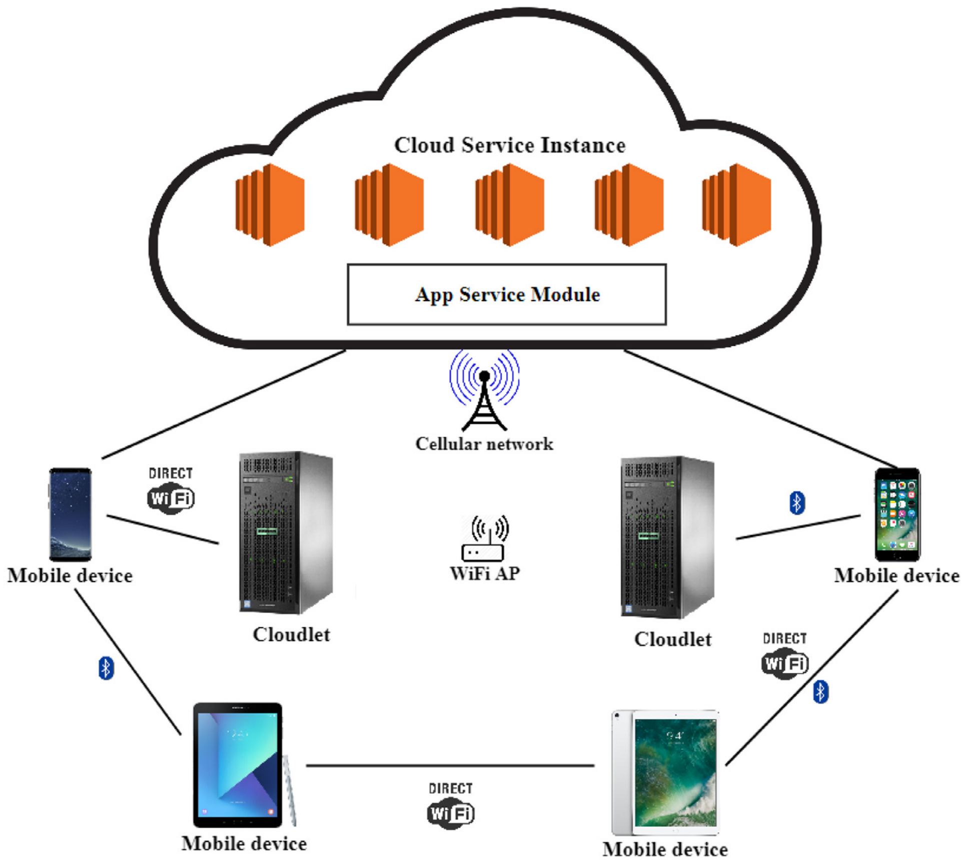


Fig. 1. An example of a local mobile device network using heterogeneous mobile cloud services.

addition, some users also own cloudlets that are publicly connectible for other mobile devices via WiFi (e.g., a café owner), and some users have subscribed to the application's cloud service with dedicated instances running in the cloud. The different types of resources, i.e. heterogeneous resources, form a shared resource pool that enables mobile cloud offloading services. However, as an opportunistic network of computing resources, the mobile users may not be willing to provide their own devices to other users for running tasks due to the limited battery or privacy concerns. Therefore, the key to attaining full benefits of offloading in HMC is incentivizing all the mobile users to commit their computing resources to the mobile cloud offloading service. For users having dedicated instances running in the cloud, such incentive mechanisms will allow them to share costs with other participants of HMC.

In this paper, we propose an incentive mechanism for heterogeneous mobile cloud offloading to encourage mobile users to participate and get appropriate rewards in return. Particularly, we design an offloading market where the mobile users who wish to offload their mobile tasks to remote resources are considered as buyers who submit offloading requests to the market and the mobile users who have redundant resources are considered as sellers that can announce the prices they are charging and compete on the market to lease their resources to the buyers for offloading. Within the HMC environment, the challenges for the designed offloading market are:

- The mobile devices can be both buyer and seller, and each device may have a different amount of computing resources since the devices are heterogeneous and can own other types of resources such as cloudlets and cloud instance subscriptions. Therefore, the execution cost for each mobile device to complete the same offloading task may be different.
- Since the objective of mobile cloud offloading is to conserve energy and execution time, the resource allocation results made by the proposed incentive mechanism need to guarantee these offloading benefits.
- As individual mobile users, sellers on the offloading market may try to gain more income by overpricing their offloading services, or lose income as a result of unfair competition and underpricing. Therefore, the proposed incentive mechanism needs to guarantee a fair market with truthfulness and individual rationality.
- Since mobile devices may leave the market before the offloaded tasks are completed, the incentive mechanism needs to introduce penalty and fault recovery mechanism to reduce service disruptions.

In order to tackle these challenges, we propose a reverse auction based incentive mechanism for the designed offloading market in the HMC. The proposed mechanism consists of two parts: an auction algorithm that is responsible for allocating the offloading tasks (buyers) to the resource sellers and deciding the payments for sellers, and the fault recovery and penalty scheme for the failed sellers. The proposed auction algorithm aims to guarantee the trustfulness and individual rationality of the auction participants. The proposed incentive mechanism is evaluated under substantial simulations with real-world workload traces and real experiments with the prototype system implemented on the Android platform. In summary, the main contributions of this work are as follows.

1. We model the task offloading service in the heterogeneous mobile clouds as an offloading market where the mobile devices requesting to offload are considered as *buyers* and the mobile

devices that offer their redundant resources for offloading are considered as *sellers*. Sellers compete on the market by bidding the different types of resources such as their mobile device resources and cloudlets or cloud subscriptions to other buyers.
2. We propose a reverse auction based incentive mechanism for allocating the buyers' requests to sellers' bids and a pricing policy to decide the payment from the buyers to the sellers. Also, a fault recovery and penalty mechanism as part of the incentive mechanism is proposed to cope with sellers who cannot complete the requests assigned.
3. The reverse auction based algorithm is designed as the online algorithm and has a low time complexity. It can guarantee the individual rationality and trustfulness of participants in the auction.
4. We implement a prototype of the incentive mechanism on top of the code offloading framework mCloud (Zhou et al., 2017) we proposed before. The incentive mechanism is then evaluated with both simulations and real experiments on the prototype.

The rest of this paper is organised as follows. We first discuss the related works for incentive mechanism in mobile cloud computing in Section 2. Then the system models and the problem formulation are given in Section 3. The design of proposed incentive mechanism and the economic property proofs are presented in Section 4, followed by a description of the prototype implementation and performance evaluation in Section 5. Finally, we give a conclusion and future works in Section 6.

2. Related work

2.1. Mobile cloud offloading

Mobile cloud offloading enables mobile devices to offload the computation intensive tasks to cloud resources via wireless media. Several research works (Liu et al., 2016; Fakoor et al., 2012; Ravi and Peddoju, 2014; Zhou et al., 2017; Hung et al., 2015; Shih et al., 2015) have proposed different techniques for offloading mobile tasks. Liu et al. (2016) proposed a multi-location offloading problem in the cloudlet-based mobile cloud system. The problem is formulated with semi-Markov decision process and solved with linear programming method. Fakoor et al. (2012) designed a mobile sensing framework to enable Mobile-Application-as-a-Service (MAaaS) through mobile cloud offloading. The framework allows creation and integration of human-centric mobile cloud applications. Ravi and Peddoju (2014) proposed a handover strategy for the offloaded mobile tasks when failures are proactively monitored. Hung et al. (2015) ported Java flow-based programming onto Android for big data Android applications to offload tasks. A framework named MobileFBP was implemented to realize the proposed system. Zhou et al. (2017) proposed a code offloading framework called mCloud, which considers a multi-tier mobile cloud system that utilizes the mobile ad-hoc network, cloudlet and public clouds as offloading resources. Shih et al. (2015) designed an elastic offloading framework based on OpenCL for mobile cloud offloading. It also considered a multi-tier offloading resource environment. Zhang et al. (2015) studied the fault tolerance in mobile cloudlet systems and proposed an offloading algorithm considering user mobility, device availability, and admission control. However, most of the mobile cloud computing works focused on the offloading techniques, with little consideration of incentivizing mobile devices to participate.

2.2. Incentive mechanisms in mobile opportunistic computing

Incentive mechanisms are essential for a computing environment that involves opportunistic and volunteering mobile device users, such as Internet-of-Things (IoT) and mobile crowdsourcing Brabham. Li and Shen (2012) analysed two main incentive methods, reputation-based systems and price-based systems, in the mobile ad-hoc network, and compared the performance of these two types in terms of cooperation incentives. Tian et al. (2017) designed the incentive mechanism based on the devices' movements, which motivates participants to move around and gather more sensing data. A greedy algorithm was proposed to solve the task allocation problem. Zhang et al. (2016) proposed a multi-market dynamic double auction mechanism, MobiAuc, for the proximity-based mobile crowd service. MobiAuc addressed the overlapping multi-group mobile device problem, and it is trustful and individually rational. Wang et al. (2016) presented a reverse auction formulation for quality-aware mobile crowdsensing system to minimize the overall expenditure of the system. A quality score is assigned to each user based on the availability, accuracy of sensor data, reputation, etc. A game-based incentive mechanism for multi-resource sharing was proposed by Gan et al. (2017) to share their idle resources in mobile social crowdsourcing. A task allocation process, profit transfer process and reputation management process are combined to achieve a trustfulness and individual rationality for the proposed incentive mechanism. Liu et al. (2017) studied the incentive mechanism for computation offloading in IoT by proposing a Stackelberg game-based approach. The approach was shown capable of reaching a unique Nash equilibrium. As we can observe from the related works, two main approaches for incentive in the voluntary computing network are auction based and game theory based. In this regard, it is worth mentioning that there are also other approaches such as Contract theory, which are being applied in this context (Zhang and Han, 2017).

2.3. Incentive mechanisms in mobile cloud computing

Only a few works have studied the incentive issue for heterogeneous mobile clouds. One closely related work is presented by Jin et al. for HMC (Jin et al., 2016). They considered the mobile devices as buyers only and the cloudlets and public clouds as sellers only. A two-stage double auction is designed to determine the winners and prices in buyers and sellers to allocate the offloaded tasks, and further select seller candidates for buyers who won more than one auction. Similarly, Xie et al. (2017) designed a distributed multi-dimensional pricing mechanism based on game theory for cloudlet-based offloading allocation and scheduling. Three types of prices such as multi-dimensional price, penalty price, and discount price are designed to motivate resource sharing. Game theoretic incentive scheme was also considered by Mir and Srirama (2018) for offloading in a pure device-to-device (D2D) setup. In line, Zhou et al. (2018) proposed a three-stage auction scheme by combining cloudlet placement and resource assignment, considering a group-buying mechanism. Similarly, a distributed auction mechanism for task offloading among mobile devices is considered by Wang et al. (2018). On the other hand, Tang et al. (2017) proposed a double auction based incentive mechanism for mobile adhoc network cloud with a single market-clearing price policy. Different user behaviour of price taking and price anticipating are analysed with a game-theoretic approach.

The existing works have not studied the incentive issue in the proposed HMC environment where all devices can be both sellers and buyers. Moreover, the computation efficiency of incentive mechanisms is vital for mobile devices and auction mechanisms as the double auction is approved to have high time complexity and are hard to implement (Myerson and Satterthwaite, 1983). Hence, we proposed a reverse auction based incentive mechanism for HMC that satisfies trustfulness and individual rationality, with low computational complexity.

3. System model and problem formulation

In this section, we first present the system model that abstracts the various characteristics of the proposed heterogeneous mobile cloud (HMC) environment. Then, based on the system models, we formulate the offloading market problem in HMC environment in the form of integer linear programming. The objective of the problem is to maximize the income of the mobile device users for sharing resources, with the consideration of offloading benefits and the unique constraints in the HMC environment.

3.1. System model

As depicted in Fig. 1, there are three types of computing resources considered in the HMC environment: Mobile devices such as smartphones and tablets, nearby cloudlets like laptops, and public cloud instances managed by the mobile app provider.

Consider there are a number of m mobile devices on the market. The mobile devices have at least one of the wireless network interfaces available, including WiFi, cellular network, Bluetooth and WiFi-direct. All devices on the market are running the same mobile cloud application. A mobile device m_i is modelled as follows.

$$m_i =< \mu_i, \theta_i, I_{wifi}, I_{cell}, I_{bt}, P_i^{active}, T_i^{avail} > \tag{1}$$

where μ_i represents the processing speed of device m_i, θ_i represents the utilization of the processor, I_{wifi}, I_{cell}, I_{bt} are the binary indicator for the wireless medium availability, P_i^{active} is the energy consumption rate when the device is active, and T_i^{avail} is the device's available time on the market.

The cloudlets are mobility-enhanced small-scale cloud data centres that are located at the edge of the Internet. Each cloudlet is considered as an always available and stable computing resource owned by one of the users on the market. The cloudlet is accessible via wireless networks. The hardware properties can be abstracted using the same model as the mobile device.

The public cloud services are provided and managed by the mobile cloud application provider via app service module. It is enabled within the app in the form of subscriptions. The subscriptions contain service description including capacity, service time, and price. The public cloud instances will be dedicated to the user for the purchased period. The subscriptions can be modelled as follows.

$$S_j^{com} =< \omega_j, T_j^{com}, p_j^{com} >, \tag{2}$$

where ω_j represents the processing speed of the cloud instance subscription S_j^{com}, T_j^{com} is the purchased duration of purchased subscription S_j^{com}, and p_j^{com} denotes the price rate of the subscription.

We consider the task offloading in HMC as an offloading market, where the mobile user can run the application entirely on his/her local mobile device or offload the computation intensive tasks to other computing resources in the network which are sharing their redundant resources. On the offloading market, the mobile users who wish to offload their computation intensive tasks are considered as buyers, while mobile users who wish to share their redundant resources are considered as sellers who compete with different prices to fulfil the buyers' offloading requests. The redundant resources can be the local resource of other mobile devices, nearby cloudlet resources owned by other users, and public cloud resources owned by the app users who have purchased them. In order to abstract the task offloading processes, the offloading request is modelled as follows.

The user who wishes to offload tasks to other resources can submit a task offloading request to the auctioneer in the HMC network. The request contains the requested resources and the demand of the resource usage including requested amount of computation and task deadline.

$$r_i =< c_i, d_i >, \tag{3}$$

where c_i represents the demanded amount of computation for the offloading task, and d_i represents the deadline of the offloading task.

Similarly, the user who wishes to sell their redundant resources on the market can submit a service offer to the auctioneer in the network. The service offer contains the specifications of the resources, and it is modelled as follows.

$$s_j =< \omega_j, t_j^s, t_j^f, b_j > \tag{4}$$

where ω_j represents the processing speed of the service offer s_j, t_j^s and t_j^f are the starting and finishing time of s_j, and b_j denotes the bid of the seller. b_j is determined by the seller, and it is supposed to be reported truthfully to his/her real valuation of the job with the help of the auction algorithm design.

At last, the cost of task execution is modelled in terms of execution time and energy consumption for the offloading tasks. Assuming there are a number of i task offloading requests and a number of j devices offering the offloading services, then the execution time for task i on the leased resource of device j is

$$\begin{aligned} T_{ij}^{comp} &= \frac{c_i}{\omega_j} + \frac{S_{in} + S_{out}}{\nu_{ij}}, \\ \omega_j &= \mu_j \theta_j, \end{aligned} \tag{5}$$

where S_{in} and S_{out} are the input data size and output data size of task i, and ν_{ij} denote the bandwidth for offloading task i onto device j. c_i is the required amount of computation of task i.

The energy consumption for task i on the leased resource of device j can be modelled as follows.

$$E_{ij}^{comp} = P_j^{active} T_{ij}^{comp} + \sigma \frac{S_{in} + S_{out}}{\nu_{ij}}, \tag{6}$$

where σ is the energy consumption rate of the wireless medium transmission, and it varies depending on the type of the wireless medium. Therefore, the overall cost of executing task i on the resource of device j is

$$C_{ij}^{comp} = \alpha T_{ij}^{comp} + \beta E_{ij}^{comp}, \tag{7}$$

where α, β are two weight factors that can be adjusted by the app providers. $\alpha + \beta = 1$.

Moreover, mobile users need to value the cost of tasks executing on their own devices before submitting an offloading requests. The local task execution cost can be modelled as follows.

$$\begin{aligned} T_{il}^{comp} &= \frac{c_i}{\mu_l}, \\ E_{il}^{comp} &= P_l^{active} T_{il}^{comp}, \\ C_{il}^{comp} &= \alpha T_{il}^{comp} + \beta E_{il}^{comp}. \end{aligned} \tag{8}$$

The subscript l is used as a sign of local execution. The time execution and energy consumption are similar to the offloading cost models, without the data transmission. The notations of the models are summarized in Table 1.

3.2. Problem formulation

Traditionally, the resource allocation phase in the auction will decide the allocation results solely based on the bids from the buyers and/or sellers. That is, the seller who offers the highest bid in the reverse auction wins. However, in the proposed offloading market for HMC, the auctioneer also needs to consider the offloading benefits for each mobile device, i.e., conserving energy and reducing execution time. Therefore, we formulate the proposed offloading market problem so as to maximize the overall revenues for all

Table 1
Notations.

Notation	Description
μ	processor speed of a device
θ	processor utilization of a device
I_{cell}, I_{wifi}, I_{bt}	binary indicator of cellular, WiFi, and Bluetooth availability of a device
v_{mn}	the network speed of the wireless link between device m and n
P_t^{active}	active energy consumption rate of a device
T^{avail}	available time of a device in the network
c	required amount of computation of an offloading task
d	deadline of an offloading task
S_{in}, S_{out}	size of the input and output data
S_j^{com}	cloud instance subscription purchased by device j
ω_j	processing speed of cloud instance subscription S_j^{com}
t_j^s, t_j^f	starting and finishing time of cloud instance subscription S_j^{com}
p_j^{com}	the price of the subscription S_j^{com}
T_{ij}^{comp}, E_{ij}^{comp}	execution time and energy consumption of task i running on device j
x_{ij}	binary decision variable for task i offloading to device j
y_j	binary decision variable for winner device j of the market
p_{mn}	payment from buyer device m to seller device n
b_j	bid provided by buyer device j

the participants in the auction, with the consideration of the offloading benefits and system constraints of HMC environment. The problem is formulated in the form of integer programming method as follows.

Given a set of offloading requests $r_i \in R$ and a set of services $s_j \in S$, the problem is how to allocate the requests to the available service offers so that each mobile device user can gain maximum revenue from sharing their redundant resources (from service providers' perspective), while the cost of executing tasks in terms of time and energy are minimized (from the task offloading perspective), since a mobile device user can be both a resource seller and a resource buyer for offloading.

We first define the binary decision variables for the allocation results. Let x_{ij} denote that request r_i is allocated to service offer s_j if $x_{ij} = 1$, otherwise if $x_{ij} = 0$. Let y_j denote that service offer s_j wins the auction if $y_j = 1$, otherwise if $y_j = 0$. In order to ensure a valid task schedule for the machines, two supporting variables are introduced. Let t_{ij}^s be a continuous variable representing the execution starting time of the task i on service j, and o_{ikm} be a binary variable representing the order of task schedule, for which $o_{ikj} = 1$ if task i is in front of task k on the service j. Then t_{ij}^f can represents the finishing time of task i on service j, where $t_{ij}^f = (t_{ij}^s + T_{ij}^{comp})x_{ij}$. The problem is formulated as follows.

$$max \sum_{j\in S} y_j \sum_{i\in R} p_{ij}x_{ij} - \sum_{i\in R}\sum_{j\in S} C_{ij}^{comp} x_{ij}$$
$$s.t. \tag{9}$$

$$x_{ij} \leqslant y_j, \forall i \in R, \forall j \in S \tag{10}$$

$$\sum_{j\in S} x_{ij} = 1, \forall i \in R \tag{11}$$

$$x_{ij}C_{ij}^{comp} + \overline{T}(x_{ij} - 1) \leqslant C_{im}^{comp}, \forall i \in R, \forall j \in S \tag{12}$$

$$\sum_{i\in R} x_{ij}T_{ij}^{comp} \leqslant y_j T_j, \forall j \in S \tag{13}$$

$$t_{ij}^f \leqslant t_i^{arrival} + d_i, \forall i \in R, \forall j \in S \tag{14}$$

$$x_{ik} + x_{ki} + o_{ikj} + o_{kij} \leqslant 3, \forall i, k \in R, \forall j \in S \tag{15}$$

$$\overline{T}(o_{ikj} - 1) \leqslant t_{kj}^f - t_{ij}^s \leqslant \overline{T} o_{ikj}, \forall i, k \in R, \forall j \in S \tag{16}$$

$$t_{ij}^s \geqslant t_j^{join} + \overline{T}(x_{ij} - 1), \forall i \in R, \forall j \in S \tag{17}$$

$$t_{ij}^s \geqslant t_i^{arrival} + \overline{T}(x_{ij} - 1), \forall i \in R, \forall j \in S \tag{18}$$

$$t_{ij}^f \leqslant t_j^{leave} + \overline{T}(x_{ij} - 1), \forall i \in R, \forall j \in S \tag{19}$$

$$x_{ij}, y_j \in \{0, 1\}, \forall i \in R, \forall j \in S \tag{20}$$

The first component in the objective function (9) represents the overall payment received by service sellers on the market, which is the sum of the winning bids, and the second component in the objective function (9) represents the overall execution cost of the task requests on the market. Constraint (10) ensures that only the devices that have won the auction can be used for offloading. Constraint (11) shows that one task can be offloaded to one and only one device for execution, that is, the task is either offloaded to a mobile device that offers its resources (device j), or executed on its own device i. Constraint (12) guarantees that the execution cost of the task on an offloaded device is less than running locally. Constraint (13) ensures the tasks allocated to the device is less than or equal to the device maximum available time for the offloading services. Constraints (14) prevent the task from finishing after its deadline. Constraints (15)–(19) ensure that there is one and only one task executing at a time on the service providing machines, and there is no overlapping on the execution orders. At last, Constraint (20) ensures the decision variables are either 0 or 1.

The optimal solution can be obtained by solving the proposed model. However, the problem is NP-hard as it can be shown that the knapsack problem can be polynomially reduced to the problem captured with the equations (9)–(20).

Theorem 1. *The proposed offloading market problem is NP-hard.*

Proof. We first show that a simplification of the proposed offloading market problem is NP-hard by giving a solution from reducing polynomially from the *Knapsack problem*. Then the simplification is relaxed to the original problem and proved to be NP-hard.

Assume that each mobile device on the market has a number of identical tasks to offload. Each mobile device as the resource seller has different processing capacity and a maximum service period limit. Therefore, the identical tasks may have different execution costs C_{ij}^{comp} on different sellers. Then this simplified offloading case can be seen as a multiple knapsack problem where the assigned tasks on a knapsack (mobile device) cannot exceed its maximum service time.

Now we relax the assumption so that each mobile device can offload heterogeneous tasks. Since there are finite number of tasks from each mobile device and a finite number of mobile devices on the offloading market, it takes polynomial time to calculate the execution cost of each task on each mobile device, and then devising the allocation of the set of offloading tasks onto mobile devices is corresponding to the simplified case. Therefore, the proposed offloading market problem, in equations (9)–(20), is NP-hard. □

We can observe that the term $y_j \Sigma_{i \in R} p_{ij} x_{ij}$ is the product of two binary variables, which is not a linear expression. This can be relaxed to a linear form by adding another binary variable and related constraints to replace the quadric term. The linearisation is as follows. First, a binary variable z_{ij} is defined as

$$z_{ij} = x_{ij} * y_j. \tag{21}$$

Then the quadric term is replaced by the z_{ij} equivalently with the following additional constraints:

$$z_{ij} \leqslant x_{ij}, \tag{22}$$

$$z_{ij} \leqslant y_j, \tag{23}$$

$$z_{ij} \geqslant x_{ij} + y_j - 1. \tag{24}$$

Therefore, the integer linear programming formulation of the proposed problem is given as:

$$\begin{aligned} max : & \sum_{i \in R} \sum_{j \in S} (p_{ij} z_{ij} - C_{ij}^{comp} x_{ij}) \\ s.t. : & (10) - (24) \end{aligned} \tag{25}$$

The optimal allocation results can only be devised for the small-scale HMC network. It is too time-consuming for the auctioneer to obtain optimal solutions when there are a large number of mobile devices and offloading requests. Therefore, an online incentive mechanism is needed to solve this problem at runtime and produce the near-optimal solutions. Moreover, since the true cost and valuation of the offloading requests for the buyers and sellers are confidential to themselves only, the only information exposed to the auctioneer is the offloading request and the bidding price of the seller to fulfil the request. Hence, the incentive mechanism should guarantee a few economic properties of the auction to force both sides to report their true valuations. Based on the demand of the proposed offloading market for HMC, the online auction mechanism should fulfil the following properties (Parsons et al., 2011).

- Computational efficiency: the auction outcome including winning sets of buyers and sellers, the mapping, and the clearing price and payment, are solved with polynomial time complexity.
- Individual rationality: no party should lose money from joining the auction. In particular, let p_j be the payment for s_j set by the auctioneer, and b_j be the bid, $p_j \geq b_j$.
- Truthfulness: All players should use the strategy of only reporting his/her true valuation. No buyers or sellers can improve their utility by reporting a bid/ask different from its true valuation.

Therefore, we propose a greedy algorithm for resource allocation and pricing in the incentive mechanism to solve the ILP model and achieve near-optimal results as well as the above-mentioned economic properties.

4. The incentive mechanism

In this section, we present a greedy reverse auction algorithm that allocates the buyers' offloading requests to the sellers' service offers, and decides the payments for the successful transactions. The proposed algorithm takes into consideration the offloading benefits as well as the constraints in the HMC environment when allocating task offloading requests. Furthermore, the proofs of the economic properties of the proposed algorithm are presented.

4.1. The greedy reverse auction

The online reverse auction in the proposed offloading market can be described as a series of auctions on a timeline that consists of discrete time epochs. Sellers' offers s_i and the buyer offloading requests r_i arrive at arbitrary times. For each s_i, it publishes the service information including its service type, capacity, bid, and available period to the auctioneer. The published information will be discarded after its available period. For each request r_i, it contains the requested amount of computation and the user nominated task completion deadline. In each auction time epoch, the greedy reverse auction-based algorithm performs two phases: resource allocation, and payment determination.

In the resource allocation phase, although there may be more than one offloading request submitted to the auctioneer in the current auction epoch, it rarely happens when two offloading requests arrive at the auctioneer at the exact same time. Therefore, all the submitted offloading requests are put in a queue based on the order of arrival. The greedy reverse auction algorithm processes the requests in the queue, by fetching each request and allocating it to the available service offers based on the task requirement, offloading benefits and environment constraints in HMC. The details of the proposed resource allocation algorithm are listed in Algorithm 1 and are explained in detail in the following paragraph. The algorithm takes the set of offloading requests R and the set of service offers S as inputs, and returns the values of the decision variables x_{ij}, y_j as outputs.

Algorithm 1 Resource Allocation algorithm (ResAlloc).

```
Input: R, S
Output: $x_{ij}, y_j$
1: Initialize $R_{sort}, S_{sort}, x_{ij}, y_j$;
2: for all $r_i \in R$ do
3:     $R_{sort} \leftarrow Sort(R_{sort}, r_i, c_i, d_i, \text{“descend”})$;
4: end for
5: for all $s_j \in S$ do
6:     $S_{sort} \leftarrow Sort(S_{sort}, s_j, \frac{b_j}{\omega_j}, \text{“ascend”})$;
7: end for
8: for all $r_i \in R_{sort}$ do
9:     for all $s_j \in S_{sort}$ do
10:        if $\frac{c_i}{\omega_j} \leqslant d_i \wedge \frac{c_i}{\omega_j} \leqslant T_j \wedge C_{ij}^{comp} \leqslant C_{im}^{comp}$ then
11:            if $y_j \neq 1$ then
12:                $s_j \leftarrow update(T_j, \frac{c_i}{\omega_j})$;
13:                $y_j = 1$;
14:                $x_{ij} = 1$;
15:                break;
16:            end if
17:        end if
18:    end for
19: end for
```

The algorithm first sorts the queue of offloading requests in the descending order of the value of $\frac{c_i}{d_i}$, which reflects the priority of the requests in terms of computation amount and the deadline of the task (step 2–4). That is, the request with large computation or urgent deadline will be processed first. Similarly, the service offers are also sorted into the list S_{sort} in the ascending order based on the value of $\frac{b_j}{\omega_j}$ (step 5–7) so that the service with the lower bid and higher processing speed will be placed in front. Then the algorithm takes the sorted requests and offers to start the auction process (step 8–19). For each request r_i at the front of the queue R_{sort}, the algorithm iterates through S_{sort} to find the winners when three conditions are satisfied (step 9–15): 1) The offloading request can be completed before its deadline ($\frac{c_i}{\omega_j} \leqslant d_i$); 2) the service can finish the offloading request before its available time ends ($\frac{c_i}{\omega_j} \leqslant T_j$); 3) the offloading benefit for request r_i is ensured ($C_{ij}^{comp} \leqslant C_{im}^{comp}$). The winner y_j will be assigned to the request x_{ij} and considered unavailable to other requests in this auction epoch (ensured in step 11). The available time T_j of service s_j is updated by substituting the execution time of r_i (step 12). At last, the algorithm returns the results of x_{ij}, y_j after all the offloading requests are processed.

In the second phase of the greedy reverse auction algorithm, the payments of the winners are determined. Traditionally, the payment can be determined by calculating the difference of the optimal result of the objective function with and without the participation of the winners. Formally, let F_{S-j} denote the optimal result of the objective function without the presence of service j. Similarly, let F_S^{-j} denote the optimal result of the objective function without considering the bid of service j, given j as the winner in the optimal allocation results of the formulation. Then the payment is calculated as:

$$p_j = F_{S-j} - F_S^{-j}. \tag{26}$$

Since, obtaining the optimal results of the objective function is not feasible at runtime, we proposed the payment algorithm based on the allocation results of the greedy auction algorithm. Similar to the optimal payment policy, the payment to the winner is the difference of the overall system utility, which is the sum of the utility values (the notion of utility is further discussed in Theorem 2) of all the winners during the current auction epoch, with and without the presence of the current winner. After all the offloading requests have been allocated to services in S by the greedy auction algorithm, the next unallocated service offer s^* in the sorted list S_{sort} is selected as the *critical service*. If all the service offers are winners, then the last winner is selected as the *critical service*. The *critical service* represents the difference of overall system utility when taking out one of the winners in S_{sort}, due to the fact that since the service offers are sorted in the descending order of the bid, the winners are in front of s^* in S_{sort}, and when taking out one of the winners, the *critical service* s^* will be the winner of the original auction. The bid of the *critical service* is called the *critical price* b^*. Then the payment to winner s_j can be calculated as $p_j = \frac{b^*}{\omega^*}\omega_j$. The payment determination algorithm is listed in Algorithm 2.

Algorithm 2 Payment Determination (PayDet).

```
Input: R, $S_{sort}$
Output: $p_j$
1: $s^* \leftarrow critical(S_{sort})$
2: $S_{win} \leftarrow winnerSelect(S_{sort})$
3: for all $s_j \in S_{win}$ do
4:     $p_j \leftarrow \frac{b^*}{\omega^*}\omega_j$
5: end for
```

Assuming there are m offloading requests and n service offers, the computational complexity of the proposed greedy reverse auction algorithm in one auction time epoch is $O(mn(m\log m + n\log n))$, where the lowest complexity of sorting of the request set and offer set are $O(m\log m)$ and $O(n\log n)$ respectively. Then assuming allocate one request to the service cost n operations at worst, and m iterations cost mn operations. Therefore the overall computational complexity of the proposed algorithm is $O(mn(m\log m + n\log n))$. The proposed greedy reverse auction algorithm in the scale of a series of D auction time epochs is listed in Algorithm 3. In each auction time epoch d_k, the algorithm collects the available offloading requests and service offers from the mobile users on the offloading market first. Then the allocation results and payments are determined by the Algorithms 1 and 2 respectively. The algorithm returns the overall allocation results X_{ij}^D, Y_j^D and payments P_j^D after $|D|$ time epochs.

Algorithm 3 Greedy reverse auction algorithm.

```
Input: D
Output: $X_{ij}^D, Y_j^D, P_j^D$
1: for all $d_k \in D$ do
2:     $R \leftarrow$ current available offloading requests
3:     $S \leftarrow$ current available resource sellers
4:     $x_{ij}^{d_k}, y_{ij}^{d_k}, S_{sort} \leftarrow ResAlloc(R, S)$
5:     $p_j^{d_k} \leftarrow PayDet(R, S_{sort})$
6: end for
```

4.2. Failure recovery and penalty policy

Since the mobile devices on the offloading market can join and leave at any time, a failure penalty policy is needed to prevent mobile device users who win an auction from deliberately failing the requests. We propose a penalty policy that only utilizes the available resources on the market and the support of the application provider.

When a mobile user who won the offloading request leaves the market and cannot complete the request, the current state of the offloading task will be checkpointed (Koo and Toueg, 1987) and is sent to the cloud instance provided by the application provider. The remainder of the failed task will be completed on the cloud instance and the application provider charges the cost of this execution to the leaving mobile device user as a penalty, and the payment for the leaving user is reversed. In this way, the offloading user will lose nothing and have the offloading request completed, while the leaving user will pay the cost of execution as a penalty.

4.3. Economic property analysis

Having proposed the reverse auction based incentive mechanism, we prove that the proposed incentive mechanism satisfies the economic properties of individual rationality and truthfulness. These properties guarantee that the participants on the offloading market will not benefit from cheating, and avoid manipulations of the offloading market.

Theorem 2. *Truthfulness: the greedy reverse auction algorithm implements a truthful auction where no participants can increase its utility by bidding a price different from its private true valuation.*

Proof. Since the true valuation of the resource provided by the sell is private, the incentive mechanism should guarantee sellers to report their true valuation so that they will not cheat to gain more by bidding a higher price than their true costs. Let b_j be the bid of seller s_j, v_j be the true valuation of the seller's cost, and u_{b_j}, u_{v_j} be the utility of the two biddings respectively. There are two cases need to be considered, which are $b_j < v_j$ and $b_j > v_j$.

First, when $b_j < v_j$, the outcomes of the auction for seller s_j can be: (A) s_j loses the auction by bidding v_j and b_j; (B) s_j wins by bidding b_j but loses by bidding v_j; and (C) s_j wins by bidding either b_j or v_j. Note that the case where s_j wins by bidding v_j and loses with bid b_j cannot occur since the sellers' offers are sorted in the ascending order of bids in Algorithm 1.

For case (A), the utilities of seller s_j by bidding b_j and v_j are both equal to 0 since the requests are allocated to the sellers with cheaper bids. That is, $u_{b_j} = u_{v_j} = 0$.

For case (B), based on the payment policy which finds the critical price b^* among all sellers' bids, there exists $\frac{b_j}{\omega_j} < \frac{b^*}{\omega^*} < \frac{v_j}{\omega_j}$. Since the utility of the winner is calculated as $u_{b_j} = p_j - v_j$, the winner s_j in this case would obtain a negative utility. Therefore, the seller s_j would choose to either lose the utility or lose the auction, which results as $u_{b_j} \leqslant 0$.

For case (C), since the sellers' offers are sorted in the ascending order of the bids, the seller would be placed closer to the end of the S_{sort} list, when bidding with b_j than v_j. However, since this has no effect on the value of the critical price b^*, the utility of $u_{b_j} = u_{v_j} = b^* - v_j$.

As a result, for all the possible auction outcomes, when the seller s_j bids lower than his/her private true valuation, the utility is less or equal to the utility of bidding the true valuation, i.e., $u_{b_j} \leqslant u_{v_j}$. It can be proved in a similar way that $u_{b_j} \leqslant u_{v_j}$ when $b_j > v_j$. Therefore, the proposed greedy reverse auction algorithm can guarantee the truthfulness of the participants. □

Theorem 3. *Individual Rationality: The proposed greedy reverse auction algorithm is individual rational that no participants would lose value from joining the offloading market.*

Proof. In order to prove the individual rationality of the proposed algorithm, we need to prove that $p_j \geq b_j$ for all the winners $s_j \in S$. According to the Algorithm 2, each winner will be paid with the value of $\frac{b^*}{\omega^*}\omega_j$, which is based on the bid of the first unallocated seller or the last seller, if all sellers are allocated. Recall that all the sellers are sorted in the ascending order of $\frac{b_j}{\omega_j}$ in Algorithm 1, then for each winner s_j we have

$$p_i = \frac{b^*}{\omega^*}\omega_j \geqslant \frac{b_j}{\omega_j}\omega_j = b_j \tag{27}$$

Therefore, for any winner in the auction, the winner's payment $p_j \geq b_j$, and the proof is complete. □

5. Performance evaluation

In this section, we present the performance evaluation of the proposed greedy reverse auction algorithm. The experiments are discussed in two parts. First, the proposed algorithm is evaluated using simulations to test the performance in terms of the social welfare, offloading benefits, the fairness of the auction results, and the truthfulness and individual rationality enforced by the algorithm. Second, we implement the prototype of the incentive mechanism on the Android operating system and test the feasibility of the mechanism in real setting experiments. For the rest of this section, we refer to our proposed incentive mechanism as *mCloudAuc*.

5.1. Simulation settings

We conduct the simulations based on a real-world trace of mobile user activities and mobile application profiles. Table 2 lists the main parameters for the simulations.

Table 2
The configuration of simulations.

Workload	Application: Optical character recognition **Task arrival rate (per second):** 0.5 **Computation required (Instructions):** Uniform(50000, 100000) **Input data size (MB):** Uniform(0.5, 10)
Mobile device	**Behaviour trace:** MIT Reality Mining traces **Processing Speed (MIPS):** Uniform(5000,200000) **Energy consumption rate (W):** Uniform(0.07,0.6) **WiFi bandwidth(MBps):** Uniform(1.0,1.2) **Bluetooth bandwidth(MBps):** Uniform(0.2,0.3) **Cellular bandwidth(MBps):** Uniform(0.8,0.9) **Energy consumption rate of wireless medium:** $\rho_{wifi} = 1.94W$, $\rho_{bt} = 0.28W$, $\rho_{cell} = 5.56W$

First, we profile an Optical Character Recognition (OCR) application[1] on an Android mobile phone to generate the task workloads for the simulations. The Android Studio performance analysis tools are used for the profiling of the application on the method level. The profile contains the input data size of the captured photos, and the inclusive CPU running time of each method for offloading to calculate the amount of computation in instructions. The task arriving rate λ at the auctioneer is 0.5 tasks per second.

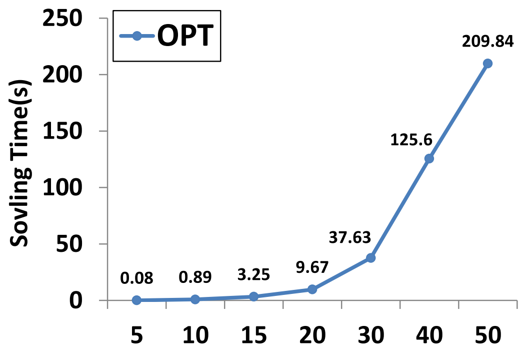
Second, the dataset MIT Reality Mining (Eagle and, Sandy) is used to extract the mobile device behaviour trace for our experiments. The dataset contains the trace of 100 mobile devices in the MIT Media Lab, which reports the location and staying duration, other devices in its proximity, idle/charging state, etc. We only consider the logs related to the cell tower associated with 'Work'. The list of such cell towers is provided by the project (Eagle et al., 2009). We first filter the datasets with the cell tower IDs to find devices connected to each of the cell towers and the corresponding connection time. Then for each set of devices found connected to each cell tower, the groups of mobile devices in each other's proximity are identified, and each group is assigned with a unique group ID. Last, the state including charge, active, and on/off are retrieved from the datasets for each device in the found groups.

In addition, the hardware properties such as processing speed and energy consumption rates are appended to the trace. The million Instructions per Second (MIPS) metric is used to quantify the processing speed, and is generated from a uniform distribution of (5000,200000). The energy consumption rates of mobile devices are obtained from the uniform distribution of (0.07, 0.6)W based on the MIPS of the processor. The power consumption rate of WiFi, Bluetooth, and cellular network are set as $\rho_{wifi} = 1.94W$, $\rho_{bt} = 0.28W$, $\rho_{cell} = 5.56W$ respectively (Balasubramanian et al., 2009). Finally, the network speed (MBps) of WiFi, Bluetooth and cellular network are set as $B_{wifi} = Uniform(1.0, 1.2)$, $B_{bt} = Uniform(0.2, 0.3)$, $B_{cell} = Uniform(0.8, 0.9)$ respectively by profiling the available network in our experiment environment.

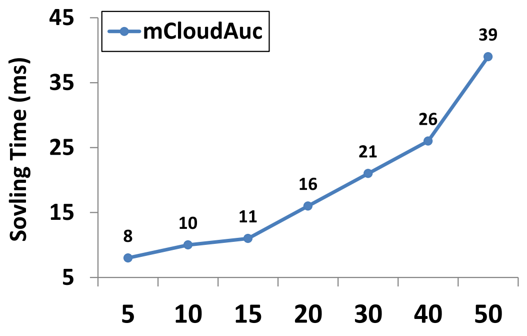
5.2. Numerical results and analysis

We conduct 5 sets of experiments for simulation to evaluate the economic properties and the performance in terms of offloading benefits for the proposed incentive mechanism.

[1] Available to download at https://github.com/rmtheis/android-ocr.



(a) Optimal solution



(b) mCloudAuc

Fig. 2. Time of devising allocation results with different number of machines.

5.2.1. Running time analysis

First, the running time of obtaining the solutions from the optimal ILP model and the proposed greedy reverse auction algorithm are compared for a different number of mobile devices on the offloading market. The optimal solution of the ILP model is obtained by using the Gurobi optimization solver (Optimization et al., 2015). The generating rate of offloading task requests for each device is randomly drawn from the uniform distribution(0.5,2). The number of task requests in each test run is managed to be around 100. All the tests are conducted on a desktop with 3,40 GHz Intel Core i7 processor and 16GB RAM.

The results of the solving time with different numbers of machines are depicted in Fig. 2. From the figure it can be observed that the optimal solution takes a significantly more amount of time for the auction process. It only takes below 40 ms for mCloudAuc (Fig. 2(b)) while obtaining results with the optimal solutions took from 80 ms up to 209 s (Fig. 2(a)) in this experiment. It can also be observed that the running time for the proposed auction algorithm increases linearly with increase in participating mobile devices, while the optimal solution increases exponentially. This result indicates a useful setting for the auction system configuration. From the economic property evaluation of mCloudAuc, discussed further in next subsection, even though optimal solution is slow, it achieves the best overall utility value. So there can be a tradeoff between choosing mCloudAuc or optimal solution. Since the auction is designed to be conducted on the cloudlet in the offloading market (Fig. 1), the service providers can set up a threshold on the number of mobile devices based on their Service-level agreements (SLAs). The auctioneer can apply optimal solution when the actual number of participants is below the threshold, or the mCloudAuc algorithm when it is above the threshold.

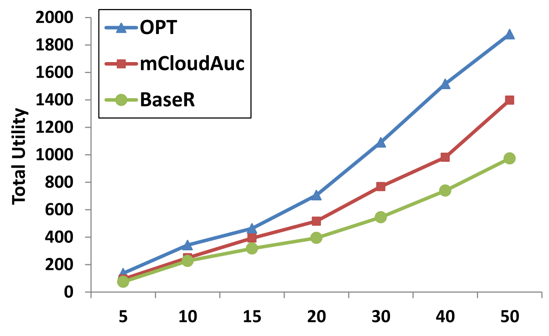
5.2.2. Economic property evaluation

In the next three sub-sections, the performance of the proposed auction algorithm is evaluated regarding the three economic properties: social welfare, individual rationality, and truthfulness. Three algorithms are compared, including 1) the optimal solution from ILP model as the benchmark, referred as **OPT**; 2) a reverse auction based incentive algorithm proposed in (Yang et al., 2012) referred as **BaseR**; 3) and the proposed incentive algorithm referred as **mCloudAuc**. The BaseR algorithm sorts bidders in a non-decreasing order of $\frac{\omega_i}{d_i}$, where ω_i represents the valuation of the task for each bidder, and d_i represents the degree (number of neighbours) of the bidder in its current computing network graph. Since the network in our case can be considered as a complete graph, the BaseR is adapted to sort by the value of ω. The adapted BaseR algorithm sorts the bidders at once at the beginning of the simulation, and the order will be fixed. The tasks are assigned on arrival with a winner from the sorted list of bidders.

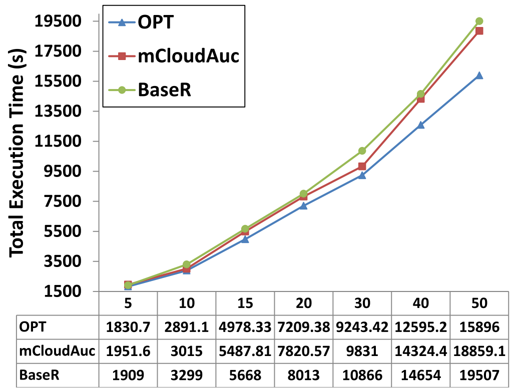
Firstly, the performance of the proposed mCloudAuc against the benchmark and baselines in terms of the overall utility of all participants are analysed with different task arrival rates and different numbers of devices on the market. In addition, the total task execution time and energy consumption of all the submitted tasks in the complete simulation are compared. The task generating rate for each mobile device is set to follow a Poisson process with $\lambda = 0.5$. The results of each test are averaged by 50 runs, and depicted in Fig. 3. Fig. 3(a) shows the sum of the utility for each participating mobile device on the offloading market. In comparison with the other algorithms, mCloudAuc generates close results to the optimal algorithm and outperforms the well-designed BaseR algorithm when the number of participants grows. In terms of task offloading benefits such as task execution and energy conservation, Fig. 3(b) and (c) show that all three algorithms perform closely to each other, while mCloudAuc can conserve more energy and execution time compared to BaseR. This is due to the dynamic ordering of both task requests and resource seller offers during each round of the auction in mCloudAuc, whereas BaseR applies a fixed order of offers when having auctions. Thus, the proposed incentive algorithm can generate resource allocation that is close to the optimal solution and maintains the offloading benefits of mobile cloud offloading at the same time.

Secondly, the price determination algorithm of mCloudAuc is evaluated to validate the individual rationality of the proposed incentive algorithm. The individual rationality of an auction mechanism ensures that participants of all parties would not lose profit from joining the auctions. Since the designed offloading market implements a reverse auction, the evaluation only focuses on the resource sellers' profits. A workload of 100 task requests randomly generated from the mobile device participants on the offloading market is tested. The bid and price paid for each resource seller and buyer pair are compared. In case a seller wins multiple task requests from different buyers, the sum of the bids and price paid are compared. The results are averaged with 50 runs, and are shown in Fig. 4. The results with 0 (e.g., machine 1,4,8) indicate that the mobile device has not won any task requests. As can be observed in Fig. 4, all the mobile devices (or resource sellers) receive payments, at least what their bids are. Therefore, the proposed mCloudAuc incentive mechanism can achieve the individual rationality as the Theorem 3 has shown.

Finally, we evaluate the truthfulness of the proposed mCloudAuc algorithm. A brief reminder that truthfulness of an auction mechanism enforces participants to only bid on their true valuation of their offers. That is, no participants would gain more profits by bidding more than their true cost. In order to test the truthfulness performance of mCloudAuc, a time-based simulation with several rounds of auctions held in between is

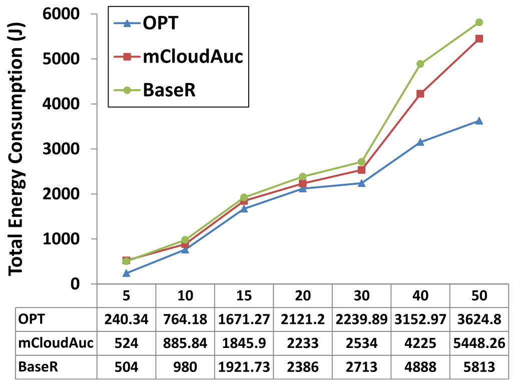


(a) Overall Utility



	5	10	15	20	30	40	50
OPT	1830.7	2891.1	4978.33	7209.38	9243.42	12595.2	15896
mCloudAuc	1951.6	3015	5487.81	7820.57	9831	14324.4	18859.1
BaseR	1909	3299	5668	8013	10866	14654	19507

(b) Overall Task Execution Time



	5	10	15	20	30	40	50
OPT	240.34	764.18	1671.27	2121.2	2239.89	3152.97	3624.8
mCloudAuc	524	885.84	1845.9	2233	2534	4225	5448.26
BaseR	504	980	1921.73	2386	2713	4888	5813

(c) Overall Device Energy Consumption

Fig. 3. Overall utility with different number of participants.

conducted to test the utility changes of the participants. Mobile device participants may join and leave at each round. Instead of generating from the distributions in Table 2, the participant's joining and leaving time used in this test are extracted from a part of the MIT Reality Mining traces. The workload consists of task requests submitted from the participants over the rounds of auctions, following a Poisson process with $\lambda = Uniform(0.2, 1)$. One mobile device participant is randomly chosen as the reference device. In each round of auctions, the selected device is manipulated to bid untruthfully from its true valuations, that is, $b_i = \delta c_i$, where δ is a weight factor to control the variation. Then the utility loss $\Delta U = U_{mCloudAuc} - U_{mCloudAuc-M}$ generated by mCloudAuc between manipulated settings and original settings is depicted in Fig. 5, with different values of δ including 0.5 (Fig. 5(a)), 0.8 (Fig. 5(b)), 1.2 (Fig. 5(c)), and 1.5 (Fig. 5(d)), which represent the selected seller underbidding his/her true cost or overbidding. Each auction epoch is set as 30 seconds.

All four subfigures show non-negative utility loss under different manipulation factor δ. The different variations for the four cases are due to the resource allocation policy and price determination policy proposed in mCloudAuc, in which the task requests and service bids are sorted at each auction round, and the payment price (or critical price) changes when the order of service bids changes with untruthful bids. The results indicate that the proposed price determination algorithm in mCloudAuc can ensure truthfulness for the participants during auctions when there are malicious participants trying to gain extra utility by underbidding or overbidding their true private valuations.

The above sub-sections evaluate the computation efficiency and economic properties of the proposed incentive mechanism mCloudAuc by simulations. The results show that mCloudAuc performs consistently with Theorems 2 and 3, and can provide near-optimal performance in terms of utility and mobile offloading benefits.

5.2.3. The implementation of mCloudAuc on Android operating system

In order to test the feasibility of the proposed mCloudAuc mechanism on real applications, we implement a prototype of the incentive mechanism framework on the Android platform (Android 5.0 with API level 21). The framework is designed in a client-server architecture where the client layer runs in Android applications, and the auctioneer server runs on a cloudlet in an HMC network (as shown in Fig. 1). In this section, the design and interaction of the modules in the framework are explained first for the client and the server, then a set of experiments are conducted on the implemented prototype using a mobile OCR application to evaluate the response delay of the proposed mCloudAuc framework running with real mobile applications. Fig. 6 illustrates an example of one auction process and the related interactions between modules on the client and the server.

The client side of the framework is implemented on the Android platform. The client framework provides the functions including a *Communication* module, an *Offloading Handler*, and a *Resource Sharing Handler*. The *Communication* module runs in a thread to deal with data encoding and decoding, commands exchange, and handling with the auction server. The *Offloading Handler* offloads mobile tasks and receives results. The *Resource Sharing*

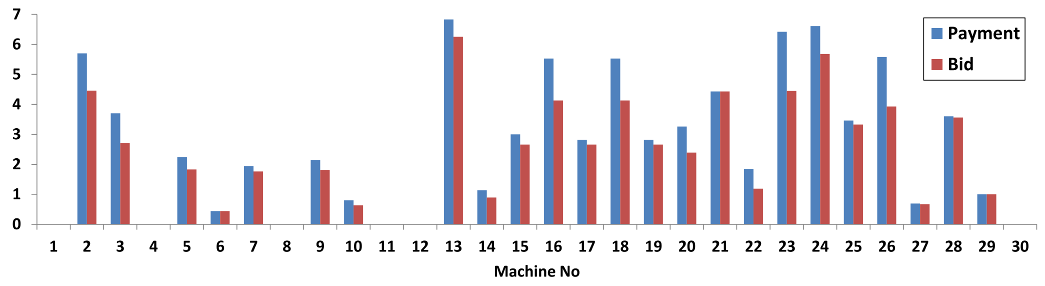
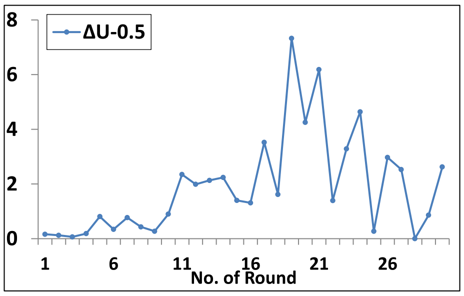
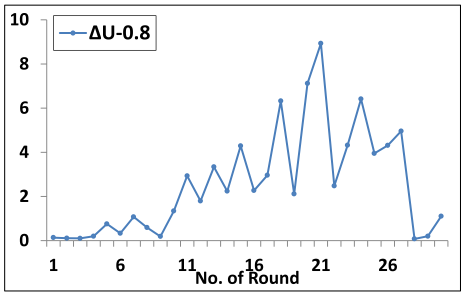


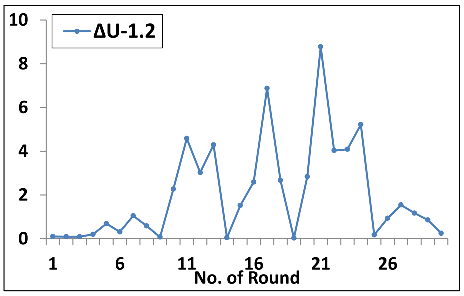
Fig. 4. Individual rationality of mCloudAuc.



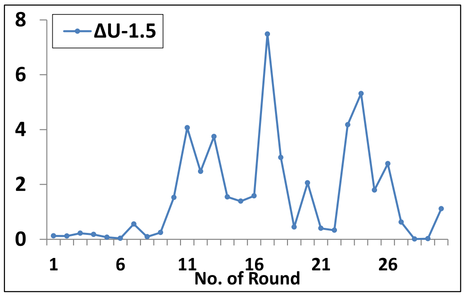
(a) $\delta = 0.5$



(b) $\delta = 0.8$



(c) $\delta = 1.2$



(d) $\delta = 1.5$

Fig. 5. Utility loss (ΔU) with different bid manipulation factor δ.

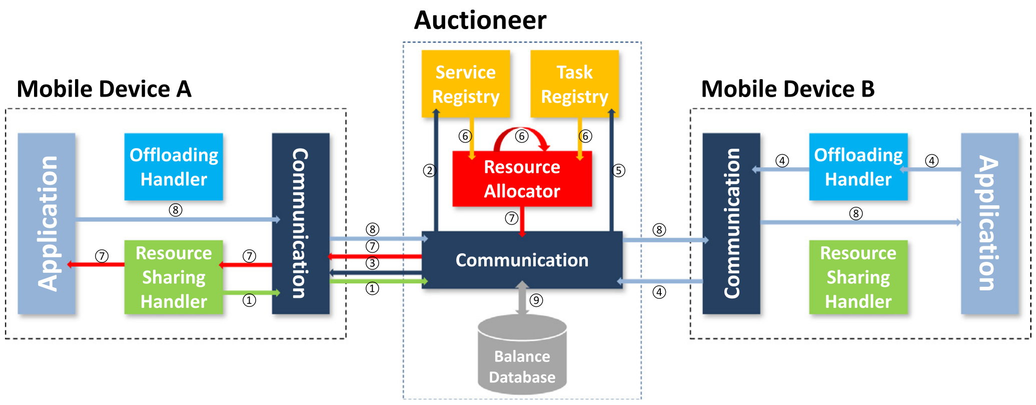
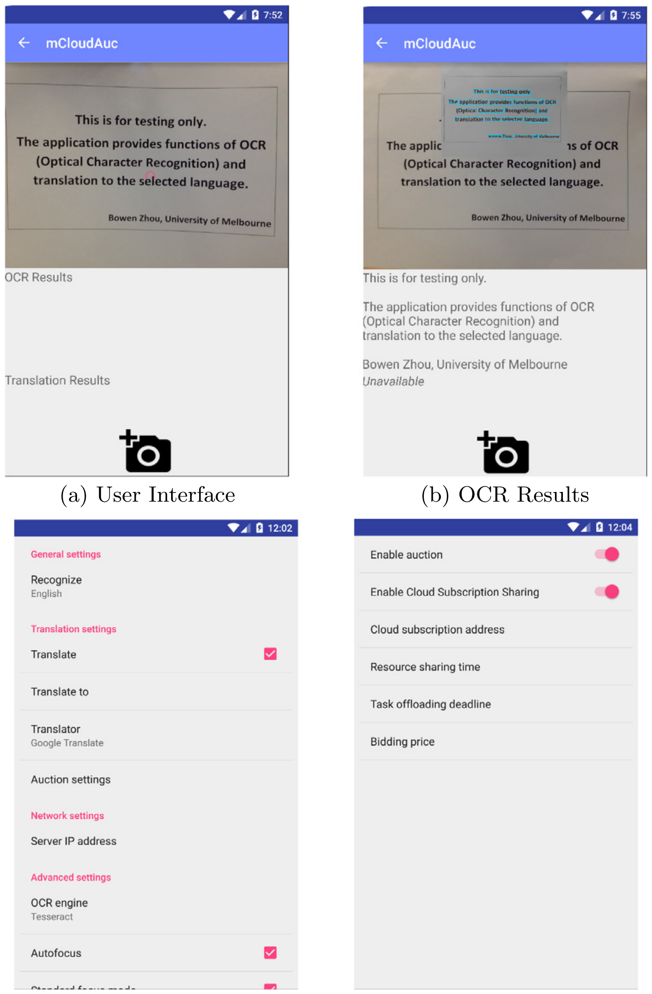


Fig. 6. An example of one auciton process and interactions of modules in the framework.

Handler registers the service offers with bids and executes the offloaded tasks from other mobile users on the shared resources.

On the auction server side of the mCloudAuc framework, five modules are implemented, including a *Communication* module, a task offloading request registry *TaskRegistry*, a service offer registry *ServiceRegistry*, a *Resource Allocator*, and a *Balance Database*. Similar to the client, the *Communication* module handles the data transmission and commands sent from the connected mobile devices, and it runs in a separate thread. The *ServiceRegistry* and *TaskRegistry* maintain the currently available service offers and task offloading requests respectively. The *Resource Allocator* implements the proposed auction logic to assign task requests to the service offers and proceeds the payments. The *Balance Database* is implemented with the NoSQL database MongoDB to provide a fast and lightweight data storage solution for managing the credit balance for users registered with the auctioneer.

The example in Fig. 6 shows the interactions of modules on mobile devices and the auctioneer, where the auctioneer allocates a task offloading request from the buyer *Device B* to a seller *Device A*. ①: *Device A* sends a service offer registration ⟨SERVICE_REGISTRATION,offer⟩ through its *Resource Sharing Handler* to the *Auctioneer* with provided processing capacity, available time and the bidding price. ②: the service offer is added to *Service Registry*. ③: *Auctioneer* sends back the confirmation with the assigned UUID ⟨REGISTRATION_SUCCESS, service_UUID⟩ to *Device A*. ④: *Device B* submits a task offloading request ⟨SUBMIT_TASK_REQUEST, task⟩ with the task and deadline through its *Offloading Handler*. ⑤: the *Auctioneer* adds the offloading request to its *Task Registry*. ⑥: When each round of the auction starts, the *Resource Allocator* takes in the list of available services and task requests to allocate the tasks using the proposed auction algorithm. ⑦: After *Auctioneer* makes the decision to allocate task request from *Device B* to service offer provided by *Device A*, it sends the task ⟨OFFLOAD_TASK, task⟩ to *Device A*. ⑧: The application on *Device A* executes the offloaded task as an AsyncTask at the background and sends the execution results ⟨SUBMIT_RESULT, result⟩ to *Device B*. If Device A is sharing its cloud subscription, the task will be sent to its dedicated instance on the cloud to execute. ⑨: The


(a) User Interface

7:52
mCloudAuc

This is for testing only.
The application provides functions of OCR (Optical Character Recognition) and translation to the selected language.
Bowen Zhou, University of Melbourne

OCR Results

Translation Results

(b) OCR Results

7:55
mCloudAuc

This is for testing only.

The application provides functions of OCR (Optical Character Recognition) and translation to the selected language.

Bowen Zhou, University of Melbourne
Unavailable

(c) Application Settings

12:02

General settings
Recognize
English

Translation settings
Translate ✓

Translate to

Translator
Google Translate

Auction settings

Network settings
Server IP address

Advanced settings
OCR engine
Tesseract

Autofocus ✓

(d) Auction Settings

12:04

Enable auction ●
Enable Cloud Subscription Sharing ●
Cloud subscription address
Resource sharing time
Task offloading deadline
Bidding price

Fig. 7. The user interface of the implemented mobile translator application.

Communication module periodically stores the payment and balance information to the *Balance Database*.

We implemented the testing application with the framework based on an open-source Android OCR (Optical Character Recognition) application.[2] Fig. 7 shows the user interface and application settings of the application. The user is able to change the settings that are related to the OCR, translation, and auction offloading in application settings. The experiment is conducted with three Android devices (one Nexus 5, one HTC EVO 3D, and one Samsung I9000) running the mobile translator application, a laptop running the auctioneer server, and an Amazon EC2 m4.xlarge instance running Genymotion on-demand Android 5.1 image as the cloud subscription. The translation application is set up to run in the background with a server socket open to listen for task offloading execution in the instance. The response delay of the auction processing time and the offloaded OCR + translation task execution time are measured using the implemented application to validate the feasibility of the proposed incentive mechanism in the real execution environment. We ran the application on HTC EVO 3D with the same set of 10 photos and obtained averaged results of 20 runs, to get statistically significant results. The results of response time and auction process time are shown in Fig. 8. Note that since different photos take different OCR processing time based on the photo quality and text content, the results in Fig. 8 are not comparable photo-wise. In Fig. 8, the blue bar on the left of each result represents the local execution time of the image OCR and translation, the green bar on the right represents the execution time of the image OCR and translation when using the offloading service, and the red bar represents the processing time of communication plus the auction processing from the auction server. As can be observed from the figure, the auction process only takes around 0.3 second, which accounts for less than 10% of the whole task response time of the offloaded tasks. Moreover, the task response time for the image OCR is reduced by over 50% on average when using the offloading service, and the additional processing time of the auction process is negligible compared to the task response time. The experiments using the implemented mCloudAuc system demonstrate that the proposed auction process added to the offloading service is short for the whole application OCR processing (less than 300 ms), and can provide significant performance enhancement to the application task response time. Therefore, the prototype implementation of mCloudAuc shows the feasibility in practice.

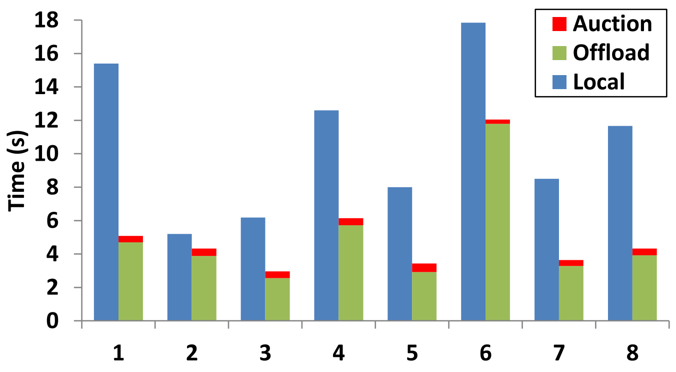


Fig. 8. Comparison of task response time between local execution and using the auction-enabled offloading service.

6. Conclusions and future work

Since the mobile cloud offloading services in HMC work in the form of opportunistic mobile network, the mobile device users may lack the incentive to share their own resources due to the battery lifetime concern. In order to encourage users to commit to the offloading services, we propose an incentive mechanism in this paper. The problem is formulated as an offloading market auction problem using ILP, where a mobile user who uses the mobile cloud offloading service can be a seller or a buyer, or both at the same time. A seller refers to a mobile user who wishes to share the redundant resources for others to offload, and a buyer refers to a mobile user who requests remote resources for offloading his/her tasks. The sellers compete by bidding different prices for the task requests on the market and receive payments from the buyers.

We propose a reverse auction-based mechanism that includes an online resource allocation algorithm and a payment determination algorithm. The resource allocation algorithm schedules the offloading task requests with the available bids based on the offloading benefits and constraints in the HMC environment. We demonstrated the computation efficiency, individual rationality and truthfulness of the proposed algorithm by both theoretical proof and simulations. The simulation results show that the proposed

[2] Available at: https://github.com/rmtheis/android-ocr.

algorithm has a near-optimal performance compared to the results obtained from the ILP models. We also implemented a prototype system of the proposed incentive mechanism on the Android platform with a mobile OCR translation application to show its feasibility in practice.

The incentive model of mobile cloud computing can also be applied to other domains such as in IoT. In IoT applications, sensor data from multiple devices/things is collected and processed at the cloud, and the control signals are sent back to the devices. While the cloud-centric IoT works well, it has significant issues with latency and data privacy. To address these things, fog computing is emerging as an alternative, where the intelligent things will take the advantage of nodes in proximity such as gateway devices, cloudlets etc. and network devices such as switches, routers etc., for processing the data locally (Srirama, 2017). To take advantage of fog computing, private participant models such as IndieFog (Chang et al., 2017) are appearing. As a future work, we will adapt our mCloudAuc in IndieFog, to provide relevant incentive mechanism as well as a fully implemented system framework for the application development.

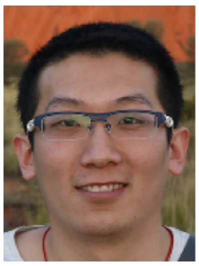
Another interesting future research direction in this domain is the aspect of considering security. Security is critical for MCC since the model deals with real mobile users and auction businesses. The paper assumed the communications among the mobile devices, cloudlets, public cloud infrastructure and the auction server are secure through standard security protocols such as HTTPS (Hyper Text Transfer Protocol Secure). The model already considered trust through truthfulness and individual rationality. However, the security study in MCC must be extended further by future works in the domain.

Acknowledgement

We thank the anonymous reviewers and Juan Carlos Duenas (Area Editor) of Journal of Systems and Software for their constructive suggestions and guidance on improving the content and quality of this paper.

References

Balasubramanian, N., Balasubramanian, A., Venkataramani, A., 2009. Energy consumption in mobile phones: a measurement study and implications for network applications. In: Proceedings of the 9th ACM SIGCOMM Conference on Internet Measurement Conference. ACM, New York, NY, USA, pp. 280–293.
Brabham, D. C., Crowdsourcing. Wiley Online Library.
Chang, C., Srirama, S.N., Buyya, R., 2017. Indie fog: an efficient fog-computing infrastructure for the internet of things. Computer 50 (9), 92–98.
comScore, 2017a. The 2017 U.S. Cross-Platform Future in Focus. Technical Report.
comScore, 2017b. Mobiles Hierarchy of Needs. Technical Report.
Eagle, N., Pentland, A.S., Lazer, D., 2009. Inferring friendship network structure by using mobile phone data. Proc. Natl. Acad. Sci. 106 (36), 15274–15278. http://www.pnas.org/content/106/36/15274.full.pdf.
Eagle, N., (Sandy) Pentland, A., 2006. Reality mining: sensing complex social systems. Pers. Ubiquitous Comput. 10 (4), 255–268.
Fakoor, R., Raj, M., Nazi, A., Di Francesco, M., Das, S.K., 2012. An integrated cloud-based framework for mobile phone sensing. In: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing. ACM, New York, NY, USA, pp. 47–52.
Gan, X., Li, Y., Wang, W., Fu, L., Wang, X., 2017. Social crowdsourcing to friends: an incentive mechanism for multi-resource sharing. IEEE J. Sel. Areas Commun. 35 (3), 795–808.
Hung, S.-H., Tzeng, T.-T., Wu, G.-D., Shieh, J.-P., 2015. A code offloading scheme for big-data processing in android applications. Software 45 (8), 1087–1101.
Jin, A.L., Song, W., Wang, P., Niyato, D., Ju, P., 2016. Auction mechanisms toward efficient resource sharing for cloudlets in mobile cloud computing. IEEE Trans. Serv. Comput. 9 (6), 895–909.
Koo, R., Toueg, S., 1987. Checkpointing and rollback-recovery for distributed systems. IEEE Trans. Softw. Eng. SE-13 (1), 23–31.
Li, Z., Shen, H., 2012. Game-theoretic analysis of cooperation incentive strategies in mobile ad hoc networks. IEEE Trans. Mob. Comput. 11 (8), 1287–1303.
Liu, Y., Lee, M.J., Zheng, Y., 2016. Adaptive multi-resource allocation for cloudlet-based mobile cloud computing system. IEEE Trans. Mob. Comput. 15 (10), 2398–2410.
Liu, Y., Xu, C., Zhan, Y., Liu, Z., Guan, J., Zhang, H., 2017. Incentive mechanism for computation offloading using edge computing: a stackelberg game approach. Comput. Netw.
Mir, T.M., Srirama, S.N., 2018. Game-theoretic incentive model for improving mobile code offloading adaptability. In: 10th IEEE International Conference on Cloud Computing Technology and Science (CloudCom 2018).
Myerson, R.B., Satterthwaite, M.A., 1983. Efficient mechanisms for bilateral trading. J. Econ. Theory 29 (2), 265–281.
Optimization, G., et al., 2015. Gurobi optimizer reference manual. www.gurobi.com.
Parsons, S., Rodriguez-Aguilar, J.A., Klein, M., 2011. Auctions and bidding: a guide for computer scientists. ACM Comput. Surv. 43 (2), 10:1–10:59.
Ravi, A., Peddoju, S.K., 2014. Handoff strategy for improving energy efficiency and cloud service availability for mobile devices. Wireless Personal Commun. 1–32.
Satyanarayanan, M., Bahl, P., Caceres, R., Davies, N., 2009. The case for vm-based cloudlets in mobile computing. Pervasive Comput. IEEE 8 (4), 14–23.
Shih, C.S., Wang, Y.H., Chang, N., 2015. Multi-tier elastic computation framework for mobile cloud computing. In: Proceedings of 2015 3rd IEEE International Conference on Mobile Cloud Computing, Services, and Engineering (MobileCloud), pp. 223–232.
Srirama, S.N., 2017. Mobile web and cloud services enabling internet of things. CSI Trans. ICT 5 (1), 109–117.
Tang, L., He, S., Li, Q., 2017. Double-sided bidding mechanism for resource sharing in mobile cloud. IEEE Trans. Veh. Technol. 66 (2), 1798–1809.
Tian, F., Liu, B., Sun, X., Zhang, X., Cao, G., Lin, G., 2017. Movement-based incentive for crowdsourcing. IEEE Trans. Veh. Technol. PP (99). 1.
Toh, C.K., 2001. Ad hoc Mobile Wireless Networks: Protocols and Systems. Pearson Education.
Wang, J., Tang, J., Yang, D., Wang, E., Xue, G., 2016. Quality-aware and fine-grained incentive mechanisms for mobile crowdsensing. In: Proceedings of 2016 IEEE 36th International Conference on Distributed Computing Systems (ICDCS), pp. 354–363.
Wang, X., Sui, Y., Wang, J., Yuen, C., Wu, W., 2018. A distributed truthful auction mechanism for task allocation in mobile cloud computing. IEEE Trans. Serv. Comput. 1–11.
Xie, K., Wang, X., Xie, G., Xie, D., Cao, J., Ji, Y., Wen, J., 2017. Distributed multi-dimensional pricing for efficient application offloading in mobile cloud computing. IEEE Trans. Serv. Comput. PP (99). 1.
Yang, D., Fang, X., Xue, G., 2012. Truthful auction for cooperative communications with revenue maximization. In: Proceedings of 2012 IEEE International Conference on Communications (ICC), pp. 4888–4892.
Zhang, H., Liu, B., Susanto, H., Xue, G., Sun, T., 2016. Incentive mechanism for proximity-based mobile crowd service systems. In: Proceedings of 2016 IEEE International Conference on Computer Communications (INFOCOM), pp. 1–9.
Zhang, Y., Han, Z., 2017. Contract Theory for Wireless Networks. Springer.
Zhang, Y., Niyato, D., Wang, P., 2015. Offloading in mobile cloudlet systems with intermittent connectivity. IEEE Trans. Mob. Comput. 14 (12), 2516–2529.
Zhou, B., Dastjerdi, A.V., Calheiros, R.N., Srirama, S.N., Buyya, R., 2017. Mcloud: a context-aware offloading framework for heterogeneous mobile cloud. IEEE Trans. Serv. Comput. 10 (5), 797–810.
Zhou, G., Wu, J., Chen, L., Jiang, G., Lam, S.-K., 2018. Efficient three-stage auction schemes for cloudlets deployment in wireless access network. Wireless Netw. 1–15.



Bowen Zhou received his BS degree from Harbin Institute of Technology, Harbin, China, in 2013 and his PhD degree in computer science from University of Melbourne, Australia, in 2018. He has been working on the computing augmentation in mobile cloud computing, and task scheduling in mobile cloud systems.

Satish Narayana Srirama is a Research Professor and the head of the Mobile & Cloud Lab at the Institute of Computer Science, University of Tartu, Estonia. He received his PhD in computer science from RWTH Aachen University, Germany in 2008. His research focuses on cloud computing, mobile web services, mobile cloud, Internet of Things, fog computing, migrating scientific computing and enterprise applications to the cloud and large scale data analytics on the cloud. He is an IEEE member, was an Associate Editor of IEEE Transactions in Cloud Computing, is an Editor of Wiley Software: Practice and Experience, a 49 year old Journal, and a program committee member of several international conferences and workshops. Dr. Srirama has co-authored over 120 refereed scientific publications in several international conferences and journals. For further information of Prof. Srirama, please visit: http://kodu.ut.ee/~srirama/.

Rajkumar Buyya is a Fellow of IEEE, Professor of Computer Science and Software Engineering and Director of the Cloud Computing and Distributed Systems (CLOUDS) Laboratory at the University of Melbourne, Australia. He is also serving as the founding CEO of Manjrasoft, a spin-off company of the University, commercializing its innovations in Cloud Computing. He served as a Future Fellow of the Australian Research Council during 2012–2016. He has authored over 525 publications and seven text books including "Mastering Cloud Computing" published by McGraw Hill, China Machine Press, and Morgan Kaufmann for Indian, Chinese and international markets respectively. He also edited several books including "Cloud Computing: Principles and Paradigms" (Wiley Press, USA, Feb 2011). He is one of the highly cited authors in computer science and software engineering worldwide. Recently, Dr. Buyya is recognized as "2016 Web of Science Highly Cited Researcher" by Thomson Reuters. Software technologies for Grid and Cloud computing developed under Dr. Buyya's leadership have gained rapid acceptance and are in use at several academic institutions and commercial enterprises in 40 countries around the world. Manjrasoft's Aneka Cloud technology developed under his leadership has received "2010 Frost & Sullivan New Product Innovation Award". Recently, Dr. Buyya received "Bharath Nirman Award" and "Mahatma Gandhi Award" along with Gold Medals for his outstanding and extraordinary achievements in Information Technology field and services rendered to promote greater friendship and India-International cooperation. He served as the founding Editor-in-Chief of the IEEE Transactions on Cloud Computing. He is currently serving as Co-Editor-in-Chief of Journal of Software: Practice and Experience, which was established over 45 years ago. For further information on Dr. Buyya, please visit his cyberhome: www.buyya.com.