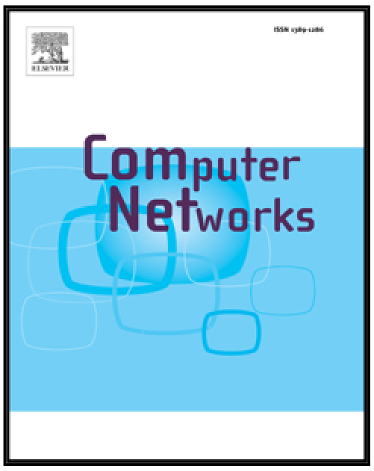


Incentive mechanism for computation offloading using edge computing: A stackelberg game approach

Yang Liu, Changqiao Xu, Yufeng Zhan, Zhixin Liu, Jianfeng Guan, Hongke Zhang

PII: S1389-1286(17)30106-8
DOI: 10.1016/j.comnet.2017.03.015
Reference: COMPNW 6130

To appear in: *Computer Networks*

Received date: 2 December 2016
Revised date: 17 February 2017
Accepted date: 21 March 2017

Incentive Mechanism for Computation Offloading using Edge Computing: A Stackelberg Game Approach

Yang Liu[a], Changqiao Xu[a,*], Yufeng Zhan[b], Zhixin Liu[c], Jianfeng Guan[a], Hongke Zhang[d]

[a]*State Key Laboratory of Networking and Switching Technology, Beijing University of Posts and Telecommunications, Beijing 100876, China*
[b]*School of Automation, Beijing Institute of Technology, Beijing 100081, China*
[c]*Institute of Electrical Engineering, Yanshan University, Qinhuangdao, Hebei 066004, China*
[d]*School of Electronic and Information Engineering, Beijing Jiaotong University, Beijing 100044, China*

Abstract

IoT-based services benefit from cloud which offers a virtually unlimited capabilities, such as storage, processing, and communication. However, the challenges are still open for mobile users to receive computation from the cloud with satisfied quality-of-service (QoS) provisioning. In this paper, we study computation offloading by using edge computing, which is a new paradigm to deliver computation to the edge of pervasive networks nearby mobile users. Without strong incentive in place, however, local edge servers may be reluctant to help offload computation. To stimulate cloud service operator and local edge server owners to participate in computation offloading, we formulate the interactions among cloud service operator and edge server owners as a Stackelberg game to maximize the utilities of cloud service operator and edge server owners by obtaining the optimal payment and computation offloading strategies. Through theoretical analysis, we show that the game is guaranteed to reach a unique Nash equilibrium. We then design two computation offloading algorithms that can quantify their efficiencies in terms of low delay and reduced complexity. Addi-

*Corresponding author
Email addresses: `liu.yang@bupt.edu.cn` (Yang Liu), `cqxu@bupt.edu.cn` (Changqiao Xu), `20120313@bit.edu.cn` (Yufeng Zhan), `lzxatuo@ysu.edu.cn` (Zhixin Liu), `jfguan@bupt.edu.cn` (Jianfeng Guan), `hkzhang@bjtu.edu.cn` (Hongke Zhang)

tionally, we extend our work by considering that edge server owners dynamically join or leave computation offloading. Numerical results show that our proposed algorithms perform well in computation offloading and efficiently stimulate edge server owners to make contribution to computation offloading.

Keywords: Computation offloading, edge computing, incentive mechanism, Stackelberg game, Nash equilibrium.

1. Introduction

The Internet of Things (IoT) is an emerging paradigm, in whch a variety of objects, which are pervasively present around us, interact and cooperate with each other to achieve common goals, making the Internet even more immersive. By making use of the enormous amount of data generated by a wide range of devices such as home appliances, surveillance cameras, vehicles and so on, the IoT has stimulated a number of attractive applications, including health care, smart cities, video surveillance, smart grid, and environmental monitoring [1, 2, 3, 4, 5, 6, 7, 8, 9, 10].

IoT-based services have gained great attention in recent years. The fact that the Internet is capable for larger data loads have had such profound impact that we have witnessed significant evolutionary eruption of the number of connected devices. According to the recent Ericsson Mobility Report, in five years, there will be 28 billion connected devices, of which 16 billion will be IoT devices [11]. With the increasing number of heterogeneous connected devices, the IoT demands significant computation and storage resources, resulting in the open question of where should those resources be placed. Cloud computing as a paradigm for big data storage and analysis makes a positive response, since it has virtually unlimited capabilities in terms of storage and processing power [12, 13, 14, 15, 16, 17, 18, 19, 20]. IoT benefits from the virtually unlimited capabilities and resources of cloud to compensate its technological constraints, e.g., storage, processing, and communication [21]. For example, multimedia content consumes more processing power and storage space, it will be more helpful to perform efficient resource management in the cloud. However, for latency-sensitive IoT services which require higher QoS, it is not feasible to communicate through the remote cloud over the Internet. To this end, edge computing which provides a way to process data at local computing servers instead of in the cloud, has been seen a promising technique to improve computation efficiency and storage utilization [22].

Edge computing is very promising for mitigating the drawbacks of current cloud computing. Rather than relying on a remote cloud, edge computing offload the computation to the nearby computing servers via wireless access by taking advantage of physical proximity. For example, 4K TV, and video streaming are the fastest growing high bandwidth applications [23, 24, 25]. In order to improve streaming and relieve network congestion, a cloud service operator (CSO) recruits edge server owners (ESOs) in close proximity of mobile users to cache the content, as illustrated in Fig. 1. In this case, the content is deployed rapidly to numerous mobile users by duplicating the content on multiple local servers and delivering the content to the nearby mobile users.

In this paper, we aim to design an efficient computation offloading scheme by using edge computing. A CSO can of course provide computation by itself. However, it is not always the best strategy because mobile users may take a long time to communicate with the public cloud through the Internet. Instead, a CSO may let local servers as depositories. In general, every server in the proximity of mobile users can perform computation offloading. The problem is even complicated that the CSO receives an offloading request and has to make a decision regarding whether or not to accept the request. Moreover, we assume that each ESO is selfish and rational, hence each owner will not perform computation offloading unless there is sufficient incentive. Each candidate owner has an available capability and will be charged a payment by the CSO if it performs computation offloading. Since ESOs may be charged different payments and have different abilities, they make different contributions to the computation offloading. These characteristics together make the development of an incentive mechanism for computation offloading by using edge computing a unique,

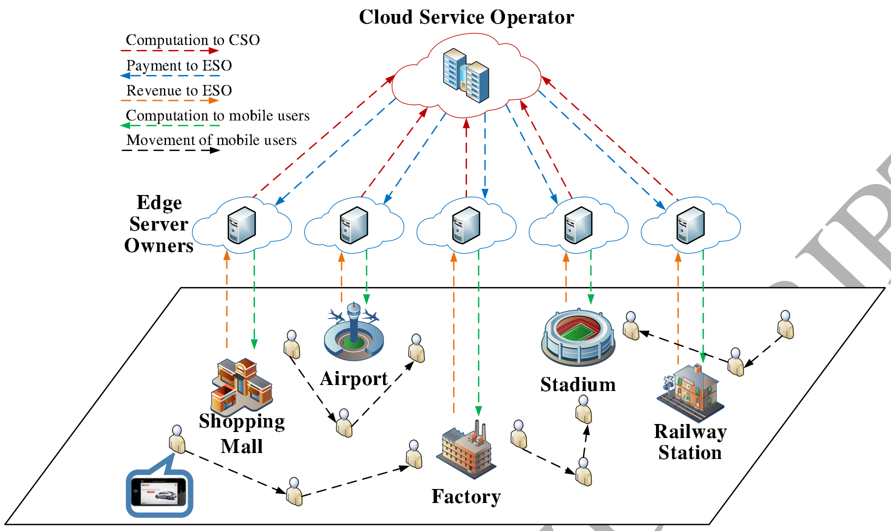


Figure 1: An example of computation offloading by using edge computing, where the red dotted curve arrow indicates the computation to the ESOs, the blue dotted curve arrow indicates the payment to the ESOs, the orange dotted curve arrow indicates the revenue to the ESOs, the green dotted curve arrow indicates the computation to mobile users, and the black dotted curve arrow indicates the movement of mobile usesrs.

interesting, and challenging problem.

We are interested in the following issues: How to offload computation efficiently? How to make both the CSO and ESOs satisfied with the offloading? To address these issues, the key questions arising in edge computiing are following: (i) how much computation should each ESO offload for CSO? and (ii) what is the corresponding payment of CSO to each ESO? We answer these questions by using the non-cooperative game theory. More specifically, we formulate the interactions among the CSO and ESOs as a two-stage Stackelberg game. In the first stage, the CSO specifies a payment profile. In the second stage, each ESO responses with the amount of its computation for offloading based on the payment. We analyze the equilibrium of the proposed Stackelberg game systematically. We design two computation offloading algorithms that have their own advantages of low delay and reduced complexity, respectively. We further consider the scenario where edge server owners may join or leave computation offloading within a computation offloading period. Through extensive simulations, we verify that the proposed algorithms have superior computation offloading performance.

We make the following contributions in this paper:

- To the best of our knowledge, this is the first work studying incentive mechanism for computation offloading by using edge computing. We formulate the interactions among the CSO and ESOs as a stackelberg game, which enables the CSO to allocate the computation based on ESOs' valuations, and jointly maximize the utilities of CSO and ESOs.
- The satisfaction nature is taken into consideration when designing the utility function of ESO, to reflect how the satisfaction varies when the ESO's demand for its own mobile users changes.
- We derive the optimal payment strategy for the CSO and the optimal computation offloading strategies for the ESOs. For the proposed Stackelberg game, we show that a Nash equilibrium exists between the CSO and ESOs, and is unique.
- We design two computation offloading algorithms based on the obtained theoretical results. They quantify their efficiencies in terms of low delay and reduced complexity, respectively.
- We consider a more general case that the ESOs may join and leave the computation offloading, and show how the equilibrium changes, which is very important to study the dynamics of computation offloading.

The remainder of the paper is organized as follows. Sec. 2 discusses related work. Sec. 3 presents the system model and formulates the problem. Sec. 4 analyzes the stackelberg equilibrium for computation offloading. Sec. 5 develops two computation offloading algorithms. We further consider a more general case under the dynamic computation offloading model in Sec. 6. Sec. 7 presents the simulation results. Finally, Sec. 8 concludes the paper.

2. Related Work

Significant study on computation offloading has been done on how to speed up mobile devices' execution time and reduce energy consumption [26, 27, 28, 29, 30, 31, 32, 33, 34]. For example, CloneCloud [26] proposes a runtime partitioning approach by combining static analysis and dynamic profiling techniques to speed up the mobile applications' execution and to reduce energy consumption. Cuervo et al. present MAUI, a system that enables fine-grained energy-aware offload of mobile code to the infrastructure [27]. MAUI decides at runtime which methods should be remotely executed, driven by an optimization engine that achieves the best energy savings possible under the mobile device's current connectivity constrains. Chen et al. study the multi-user computation offloading problem in a single-channel wireless setting, such that each user has to make a decision to offload or not [28]. Yang et al. study the multi-user computation partitioning to optimize the partition of a data stream application such that the maximum throughput is achieved [29]. Mao et al. investigate a green mobile edge computing system with energy harvest devices and propose a dynamic computation offloading algorithm based on Lyapunov optimization, which jointly decides the offloading decision, the CPU-cycle frequencies for mobile execution, and the transmit power for computation offloading [30]. However, they consider the paradigm to improve the capability of mobile services by migrating heavy computation tasks to powerful servers in cloud. Different from these works, we study the case where computation is offloaded to the local edge servers, and pay attention to the economic interactions among the cloud service operator and local edge servers.

As a novel paradigm, edge computing is also classifiable as fog computing, mobile edge computing, cloudlet, and so on [35, 36, 37, 38]. Only a few approaches are proposed recently [39, 40]. For example, Hou et al. present vehicular fog computing (VFC) which employs end-user clients and near-user edge devices to carry out communication and computation, by well utilizing each vehicle's communication and computational resources [39]. Jalali et al. compare the energy consumption of applications by using centralized data centers in cloud computing with which using nano data centers (nDCs) in fog computing, and they indicate that nDCs can achieve energy savings depending on the type of access network, the ratio of active time to idle time, and the type of applications [40]. However, none of them are developed for computation offloading. In general, computation offloading using edge computing is a less-studied area with limited existing solutions.

Game theory is the study of mathematical models of conflict and cooperation between intelligent rational decision-makers [41]. Recently, Stackelberg game has been used in studing the interactions among those decision-makers [42, 43, 44, 45, 46]. For example, Wang et al. propose a distributed game-theoretic source selection and power control scheme with low complexity that improves the transmission quality with delay constraints [42]. Yu et al. propose a general Wi-Fi bussiness model for public Wi-Fi hotspots deployed by venue owners, where venue owners generate revenue from providing both the premium Wi-Fi access and the advertising sponsored Wi-Fi access to mobile users [45]. Yang et al. study the problem of energy charging using a robust Stackelberg game approach in a power system including an aggregator and multiple electric vehicles with demand uncertainty [46]. However, the use of Stackelberg game from the above work does not fit well with the requirement of computation offloading using edge computing, where the utility function has to be customized. To the best of our knowledge, our work is the first one to use Stackelberg game to model the ESOs' interactions in computation offloading. Moreover, we take an ESO's satisfaction which models the satisfaction of a user by characterize the difference between ideal demand and actual consumption into consideration when designing the utility function of the ESO. We also design two algorithms of the game with different advantages to handle the delay and complexity between the CSO and ESOs.

3. System Model And Problem Formulation

We consider a CSO intends to offload its computation to a set $\mathcal{N} = \{1, \cdots, N\}$ of ESOs. Let X_i denote the total computation of ESO i, $\forall i \in \mathcal{N}$, and X_0 denote the computation that cannot be offloaded to any ESO, we assume X_0 is fixed in our work.

3.1. CSO Modeling

Let $x_i \in [0, X_i]$ denote the computation offloaded to ESO i, and y_i denote the CSO's payment to ESO i. The computation offloading profile and payment profile are, respectively, $\mathbf{x} = (x_1, \cdots, x_N)$, and $\mathbf{y} = (y_1, \cdots, y_N)$. Given $\mathbf{x}$ and $\mathbf{y}$, the CSO's remaining un-offloaded computation is

$$f(\mathbf{x}) = X_0 + \sum_{i=1}^{N}(X_i - x_i), \tag{1}$$

and CSO's computation without computation offloading is

$$f(\mathbf{0}) = X_0 + \sum_{i=1}^{N} X_i. \tag{2}$$

The CSO's total cost, including both the cost for maintaining the un-offloaded computation and the payment to ESOs, is

$$\begin{aligned} C(\mathbf{x}) =& C(f(\mathbf{x})) + \sum_{i=1}^{N} y_i \\ =& e^{\alpha f(\mathbf{x})} + \sum_{i=1}^{N} y_i, \end{aligned} \tag{3}$$

where $e^{\alpha f(\mathbf{x})}$ reflects the CSO's diminishing return of cost on participating ESOs. The CSO's cost without computation offloading is $C(f(\mathbf{0}))$. We define the CSO's utility as the cost reduction from computation offloading, denoted by

$$\begin{aligned} U =& C(f(\mathbf{0})) - C(\mathbf{x}) \\ =& C(f(\mathbf{0})) - C(f(\mathbf{x})) - \sum_{i=1}^{N} y_i. \end{aligned} \tag{4}$$

3.2. ESO Modeling

Each edge server is owned by an ESO, which primarily serves its own users. Thus, each ESO must take the demand of its own users into account, when deciding whether and how to offload compuation for the CSO.

Let r_i denote the revenue for one unit of ESO i's computation demand from its own users, and c_i denote the cost by offloading one unit of its computation. Then, ESO i's profit is

$$P_i(x_i) = (r_i - c_i) \cdot (X_i - x_i + g(x_i)) + y_i - c_i \cdot x_i, \tag{5}$$

where $X_i - x_i$ is the computation left for serving its own mobile users, $g(x_i) = 1 - e^{\beta_i(1 - \frac{X_i - x_i}{d_i})}$ is the satisfaction with the remaining computation, where d_i is the ideal computation demand for the mobile users for ESO i. The satisfaction function we define quantitatively models the satisfaction of a user by characterizing the difference between ideal demand and actual consumption. More specifically, if the actual computation is less than the demand, the mobile users are not satisfied, thus $g(x_i) < 0$. The value of the function decreases faster as the actual computation decreases. If the actual computation is larger than the user's demand, on the other hand, the users are satisfied, thus $g(x_i) > 0$. However, the mobile users cannot be satisfied forever when they use more computation, resulting in the slow increase of the function value. When the actual computation equals the users' demand, we obtain that $g(x_i) = 0$. β charaterizes the priority of computation offloading for the ESOs. An ESO with a larger β indicate that the ESO has a higher priority computation demand for its own mobile users. In contrast, an ESO with a smaller β has a low priority for computation demand. An illustration of computation left for serving its own mobile users with different βs and ds is shown in Figs. 2 and 3. Since the demand of computation for a mall is larger than that for an airport, the airport is more satisfied than the mall when providing the same amount of computation. On the other hand, the mall is more satisfied by giving the same amount of computation, if the demand priority of mall is higher than that of airport. In other words, we can see that $g(x_i)$ can be used to characterize different ESOs by

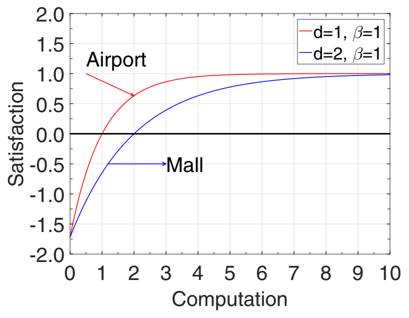


Figure 2: Satisfaction with different demands.

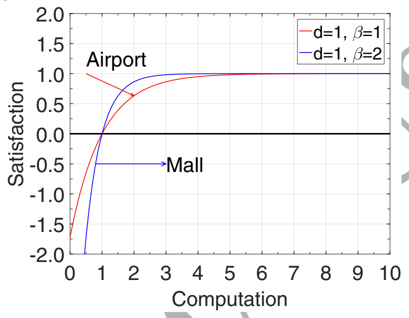


Figure 3: Satisfaction with different βs.

200 adjusting the parameters β and d. The properties of satisfaction function are summarized in the following lemma.

Lemma 1. *$g(x_i)$ satisfies $g(x_i) > 0, g^{'}(x_i) > 0, g^{''}(x_i) < 0$ if $X_i - x_i > d_i$, and $g(x_i) < 0, g^{'}(x_i) > 0, g^{''}(x_i) > 0$ if $X_i - x_i < d_i$, and $g(x_i) = 0$ if $X_i - x_i = d_i$.*

$y_i - c_i \cdot x_i$ is the profit from helping the CSO, including the CSO's payment 205 and the cost. We assume y_i is linear to x_i, which is denoted by

$$y_i = p_i \cdot x_i, \tag{6}$$

where p_i is the payment for one unit of its computation offloaded to ESO i. The ESO i's utility is the profit improvement by performing computation offloading for the CSO, which is denoted by

$$\begin{aligned}
V_i(x_i) =& P_i(x_i) - P_i(0) \\
=& (r_i - c_i) \cdot (X_i - x_i + 1 - e^{\beta_i(1-\frac{X_i - x_i}{d_i})}) + p_i \cdot x_i \\
& - c_i \cdot x_i - (r_i - c_i) \cdot (X_i + 1 - e^{\beta_i(1-\frac{X_i}{d_i})}) \\
=& (r_i - c_i) \cdot (-e^{\beta_i(1-\frac{X_i - x_i}{d_i})} - x_i + e^{\beta_i(-\frac{X_i}{d_i})}) \\
& + p_i \cdot x_i - c_i \cdot x_i.
\end{aligned} \tag{7}$$

3.3. Problem Formulation

We formulate the interactions among the CSO and ESOs as a two-stage stackelberg game to obtain the optimal payment and computation offloading strategies. In the first stage, the CSO specifies a payment profile $\mathbf{p} = (p_1, \cdots, p_N)$. In the second stage, each ESO responses with the amount of its computation for offloading based on the payment, denoted by $\mathbf{x} = (x_1(p_1), \cdots, x_N(p_N))$.

First, we study the ESOs' optimal decisions in the second stage, given the payment specified by the CSO in the first stage. Formally, the decision problem for ESO i is

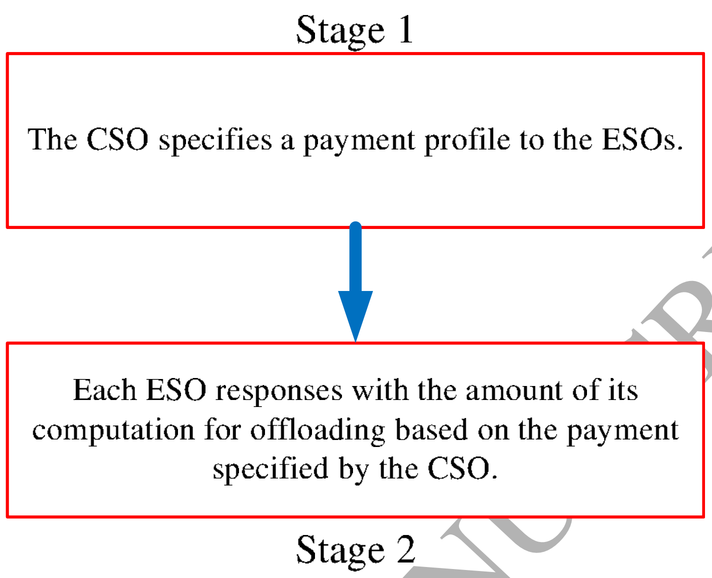
$$\begin{aligned}
\max \quad & V_i(x_i) \\
s.t. \quad & x_i \in [0, X_i],
\end{aligned} \tag{8}$$

where V_i is ESO i's utility.

Then we study the CSO's best decision in the first stage, based on its prediction of every ESO's optimal response in the second stage. The decision problem for the CSO is

$$\begin{aligned}
\max \quad & U(\mathbf{x}, \mathbf{p}) \\
s.t. \quad & p_i \geq 0, \forall i = 1, \cdots, N, \\
& x_i(p_i) \leq X_i, \forall i = 1, \cdots, N,
\end{aligned} \tag{9}$$

where $U(\mathbf{x}, \mathbf{p})$ is the utility of CSO in Eq. (4).



Stage 1
The CSO specifies a payment profile to the ESOs.
Each ESO responses with the amount of its computation for offloading based on the payment specified by the CSO.
Stage 2

Figure 4: Two-Stage Stackelberg Game.

4. Nash Equilibrium Analysis for Computation Offloading

To stimulate the ESOs to perform compuation offloading, we formulate the interactions between the CSO and ESOs as a two-stage Stackelberg game based on the non-cooperative game theory as illustrated in Fig. 4, where the CSO acts as a leader specifying the payments to ESOs, and then each ESO acts as a follower determining the computation offloaded for CSO.

We show that our problem can achieve Nash equilibrium [47]. At the Nash equilibrium, the CSO maximizes its utility based on the optimal strategies of the ESOs, each ESO is able to maximize its utility by selecting the optimal strategy. Therefore, the optimal strategy is the equilibrium strategy for each ESO. The definition of Nash equilibrium corresponding to the proposed Stackelberg game is given below.

Denote $x_i^\star$ as the optimal strategy of ESO i identified by solving problem (8), and $\mathbf{x}^\star$ as the strategy profile including all the ESOs' optimal strategies, i.e., $\mathbf{x}^\star = [x_1^\star, \cdots, x_N^\star]$. In addition, denote $p_i^\star$ as the optimal payment for ESO

i, and $\mathbf{p}^\star$ as the payment profile for all the ESOs. A Nash equilibrium is defined as such a strategy profile $\{\mathbf{x}^*, \mathbf{p}^*\}$ such that none of the ESO can improve its payment by unilitery deviating.

Definition 1. *A profile of strategies ($\mathbf{x}^\star$, $\mathbf{p}^\star$) is a Nash equilibrium for the Stackelberg game if the solution of the following optimization problem is given below:*

$$\begin{aligned} \max \quad & (\mathbf{x}^\star, \mathbf{p}^\star) = \arg\max_{\mathbf{p}\in\Omega} U(\mathbf{x}^\star, \mathbf{p}) \\ s.t. \quad & x_i^\star = \arg\max_{x_i \in \mathbf{\Psi}_i} V_i, \forall i = 1, \cdots, N, \end{aligned} \tag{10}$$

where the strategy set of ESO i is denoted as

$$\Psi_i = \{x_i | x_i \in [0, X_i]\}, \forall i = 1, \cdots, N, \tag{11}$$

and the feasible payment set of CSO is denoted as

$$\Omega = \{\mathbf{p} | p_i \in [\underline{p_i}, \overline{p_i}], \forall i = 1, \cdots, N\}, \tag{12}$$

where $\underline{p_i}$ and $\overline{p_i}$ are the lower bound and uppper bound of p_i, $\forall i = 1, \cdots, N$.

4.1. The Existence and Uniqueness of Nash Equilibrium

Lemma 2. *The strategy set of each ESO is nonempty, convex, and compact.*

Proof. The first- and second-order derivatives of V_i to x_i are, respectively,

$$\frac{\partial V_i}{\partial x_i} = -(r_i - c_i) \cdot [1 + \frac{\beta_i}{d_i} e^{\beta_i(1 - \frac{X_i - x_i}{d_i})}] + (p_i - c_i), \tag{13}$$

$$\frac{\partial^2 V_i}{\partial x_i^2} = -\frac{\beta_i^2}{d_i^2} \cdot (r_i - c_i) \cdot e^{\beta_i(1 - \frac{X_i - x_i}{d_i})}. \tag{14}$$

By taking Eq. (13) to zero, we obtain the best response as follows:

$$x_i^\star(p_i) = X_i - d_i(1 - \frac{1}{\beta_i} ln \frac{d_i}{\beta_i} \frac{p_i - r_i}{r_i - c_i}). \tag{15}$$

We regard ESO i's payment as its payment threshold by setting $x_i = 0$ and $x_i = X_i$, denoted by

$$\underline{p_i} = r_i + (r_i - c_i) \cdot \frac{\beta_i}{d_i} e^{\beta_i(1 - \frac{X_i}{d_i})}, \tag{16}$$

$$\overline{p_i} = r_i + (r_i - c_i) \cdot \frac{\beta_i}{d_i} e^{\beta_i}. \tag{17}$$

Since the computation and the payment are searched in limited regions $\Psi_i, \forall i = 1, \cdots, N$, and Ω, this lemma holds. □

Lemma 3. *Given p_i, ESO i's optimal offloaded computation satisfies*

$$x_i^\star(p_i) = \begin{cases} 0, & \textit{if } p_i \le \underline{p_i} \\ X_i - d_i(1 - \frac{1}{\beta_i} ln \frac{d_i}{\beta_i} \frac{p_i - r_i}{r_i - c_i}), & \textit{if } \underline{p_i} < p_i < \overline{p_i} \\ X_i. & \textit{if } p_i \ge \overline{p_i} \end{cases} \tag{18}$$

Proof. If $p_i < \bar{p}_i$, $\frac{\partial V_i}{\partial x_i} < 0$, thus the optimal payment for ESO i is 0. Additionally, if $p_i > r_i$, it means that the payment for one unit of its computation offloaded to ESO i is always larger than the revenue for one unit of its computation demand from its own users, thus the total computation is used by helping the CSO offload computation. Finally, if $\underline{p_i} < p_i < \overline{p_i}$, the ESO i's optimal offloaded computation is given by Eq. (15). □

Lemma 4. *Each ESO has a unique optimal strategy once informed of the CSO's strategy.*

Proof. Intuitively, if the CSO proposes a large enough payment p_i (i.e., $p_i > \overline{p_i}$), ESO i will sell all of its computation to the CSO. On the other hand, if the payment p_i is smaller than the threshold $\underline{p_i}$, ESO i will sell none of its computation to the CSO.

When $\underline{p_i} < p_i < \overline{p_i}$, we have

$$\frac{\partial x_i^\star}{\partial p_i} = \frac{d_i}{\beta_i(p_i - r_i)}, \tag{19}$$

$$\frac{\partial^2 x_i^\star}{\partial p_i^2} = \frac{-d_i}{\beta_i(p_i - r_i)^2}. \tag{20}$$

We can find that $\frac{\partial x_i^\star}{\partial p_i} > 0$, which implies that the higher payment the CSO offers, the more computation the ESO dedicates to computation offloading. Obviously, the value of Eq. (20) is always negative. Eqs. (19) and (20) imply that $x_i^\star$ is an increasing concave down function of p_i.

Moreover, since the strategy set Ψ given in Eq. (11) is concave, $V_i(x_i)$ is strictly concave in Ψ_i, thus the optimal strategy in terms of (15) is guaranteed to be optimal and unique. Lemma 4 is proved. □

4.2. CSO's Decision

The first-order partial derivative of U to p_i is

$$\begin{aligned}\frac{\partial U}{\partial p_i} =& C^{'}(f(\mathbf{x}^\star)) \cdot \frac{\partial x_i^*}{\partial p_i} - p_i \cdot \frac{\partial x_i^*}{\partial p_i} - x_i^* \\ =& e^{f(x^\star)} \cdot \frac{d_i}{\beta_i(p_i - r_i)} - p_i \cdot \frac{d_i}{\beta_i(p_i - r_i)} - x_i^\star.\end{aligned} \tag{21}$$

Lemma 5. *For optimal payment profile* $\mathbf{p}^*$*, it satisfies that*

$$\begin{aligned}&\underline{p_i} < p_i^\star < \overline{p_i}, \forall i = 1, \cdots, N, \\ &p_i^\star \le C^{'}(f(\mathbf{x}^\star)), \forall i = 1, \cdots, N.\end{aligned} \tag{22}$$

Proof. By Lemma 1, if $p_i \le \underline{p_i}$ or $p_i \ge \overline{p_i}$, $\frac{\partial x_i^*}{\partial p_i} = 0$, thus we have $\frac{\partial \mathbf{U}}{\partial p_i} = 0$ if $p_i \le \underline{p_i}$ and $\frac{\partial \mathbf{U}}{\partial p_i} = -X_i$ if $p_i \ge \overline{p_i}$, it implies that any payment lower than $\underline{p_i}$ is indifferent to the CSO, and any payment higher than $\overline{p_i}$, the utility is always decreasing. Therefore, the first inequation holds.

For the second inequation, we prove by contradiction. If $p_i^\star > e^{f(x^\star)}$, we have $\frac{\partial \mathbf{U}}{\partial p_i} < 0$, it implies that a payment higher than $p_i^\star$ can achieve larger utility of the CSO, therefore the second inequation holds. □

Intuitively, if $p_i \le \underline{p_i}$, ESO i always returns zero amount of its computation for offloading, and thus any payment p_i lower than $\underline{p_i}$ is indifferent to the CSO. If $p_i \ge \overline{p_i}$, ESO i always returns all of its computation for offloading, and thus any payment higher than $\overline{p_i}$ cannot bring more benefit for the CSO, but will definitely lead to a higher payment.

Lemma 6. *The CSO has a unique optimal strategy given the best strategies of all the ESOs.*

295 *Proof.* We take the second-order derivative of $U(\mathbf{x}^*, \mathbf{p}^*)$ with respect to p_i in the form of the Hessian matrix of $U(\mathbf{x}^*, \mathbf{p}^*)$. We have

$$\begin{aligned}\frac{\partial^2 U}{\partial p_i^2} =& - C^{''}(f(\mathbf{x}^\star)) \cdot (\frac{\partial x_i^*}{\partial p_i})^2 - 2 \cdot \frac{\partial x_i^*}{\partial p_i} \\ &+ (C^{'}(f(\mathbf{x}^\star)) - p_i)\frac{\partial^2 x_i^*}{\partial p_i^2},\end{aligned} \tag{23}$$

$$\frac{\partial^2 U}{\partial p_i \partial p_j} = -C^{''}(f(\mathbf{x}^\star)) \cdot \frac{\partial x_i^*}{\partial p_i}\frac{\partial x_j^*}{\partial p_j}. \tag{24}$$

We define

$$\begin{aligned}H_1 =& \mathbf{diag}((C^{'}(f(\mathbf{x}^\star)) - p_1)\frac{\partial^2 x_1^*}{\partial p_1^2} - 2 \cdot \frac{\partial x_1^*}{\partial p_1}, \cdots, \\ &(C^{'}(f(\mathbf{x}^\star)) - p_N)\frac{\partial^2 x_N^*}{\partial p_N^2} - 2 \cdot \frac{\partial x_N^*}{\partial p_N}),\end{aligned} \tag{25}$$

$$H_2 = -C^{''}(f(\mathbf{x}^\star)) \begin{bmatrix} \frac{\partial x_1^*}{\partial p_1} \\ \frac{\partial x_2^*}{\partial p_2} \\ \vdots \\ \frac{\partial x_N^*}{\partial p_N} \end{bmatrix} \begin{bmatrix} \frac{\partial x_1^*}{\partial p_1} \frac{\partial x_2^*}{\partial p_2} \cdots \frac{\partial x_N^*}{\partial p_N} \end{bmatrix}, \tag{26}$$

$$H = H_1 + H_2, \tag{27}$$

300

$$v^T H v = v^T H_1 v + v^T H_2 v, \tag{28}$$

where v is a non-zero column vector.

Since $p_i \le C^{'}(f(\mathbf{x}^\star))$, $\frac{\partial^2 x_i^\star}{\partial p_i^2} \le 0$, $\frac{\partial x_i^\star}{\partial p_i} \ge 0$, we have

$$v^T H_1 v \le 0. \tag{29}$$

Since $C^{''}(f(\mathbf{x}^\star)) \ge 0$, we have

$$v^T H_2 v = -C^{''}(f(\mathbf{x}^\star)) \cdot \sum_{i=1}^{N} (\frac{\partial x_i^*}{\partial p_i})^2 \le 0. \tag{30}$$

Thus, we have

$$v^T H v \le 0. \tag{31}$$

Therefore, $U(\mathbf{x}, \mathbf{p})$ is strictly concave. Then we can obtain that the problem (10) is a convex optimization problem. Meanwhile, the optimal strategy $\mathbf{p}^\star$ is unique, because the Hessian matrix is strict negative definiteness. Therefore, Lemma 5 is proved. $\square$

The above lemmas together prove the following theorem.

Theorem 1. *A unique Nash Equilibrium exists between the CSO and ESOs in our proposed Stackelberg game.*

Theorem 2. *The payment p_i for ESO i satisfies the following condition:*

$$C^{'}(f(\mathbf{x}^\star)) = p_i + x_i^\star \cdot \frac{\partial p_i}{\partial x_i}, \forall i \in \mathcal{N}. \tag{32}$$

According to Eq. (3), the left hand side of Eq. (32) is the marginal utility of CSO, while the right hand side of Eq. (32) is the marginal payment to ESO i, i.e., the increase of ESO i's payment induced by offloading one unit of computation to ESO i. More specifically, to increase the computation by one, the CSO has to increase the payment by $\frac{\partial p_i}{\partial x_i}$, introducing an additional payment $\frac{\partial p_i}{\partial x_i}$ for the existing $x_i^\star$ units of offloaded computation, and a new payment p_i for the coming one unit of offloaded computation. Eq. (32) suggests that the marginal utility of CSO equals to the marginal payment to ESO i. Intuitively, if the marginal utility of CSO is larger (or smaller) than the marginal payment to ESO i, the CSO can improve its payment by offloading more (or less) computation to ESO i through increasing (or decreasing) p_i.

5. Mechanism Design

In this section, we propose to design two algorithms which have advantages of low delay and reduced complexity, respectively, as illustrated in Fig. 5.

5.1. One Round Stackelberg Game

One round Stackelberg game is described in Algorithm 1. Since the above algorithm only needs one round to obtain the optimal payment and computation

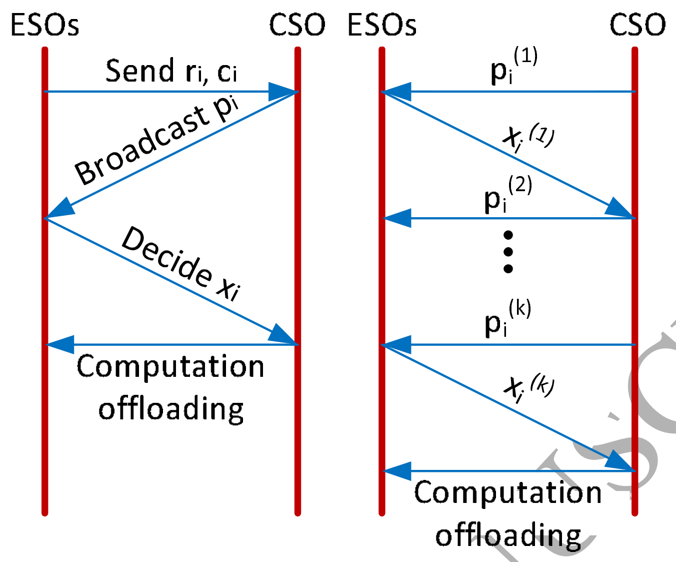


Figure 5: (a) One Round Stackelberg Game. (b) Multi-round Stackelberg Game.

of ESOs, it achieves small computation time. Thus, for latency-sensitive service [48, 49], Algorithm 1 is preferred.

5.2. Multi-round Stackelberg Game

While the above algorithm is time-saving, it requires the CSO to directly compute the optimal payment introduced in Sec. 4, which consumes a large amount of computing power of ESOs. In contrast, for Multi-round Stackelberg Game as illustrated in Algorithm 2, the ESO and the CSOs go through multiple rounds to reach the equilibrium. Thus, the Multi-round Stackelberg Game greatly reduces the computation on the ESO. The ESO only needs to compare the upper bound and lower bound of payment. In this case, Multi-round Stackelberg Game should be preferred by the edge server owners with less computing power.

We propose Multi-round Stackelberg Game in Algorithm 2. More specifically, we set four payments, denoted as $\underline{p} = 0, \overline{p} = \max\{\overline{p}_i | i \in N\}, \hat{p} = \frac{\underline{p}+\overline{p}}{2}$, and $\check{p} = \frac{\hat{p}+\overline{p}}{2}$. The CSO sets initial payments $\hat{p}_i, \check{p}_i, \forall i \in \mathcal{N}$, and sends them

Algorithm 1 One Round Stackelberg Game

```
Input: $r_i$, which is the revenue for one unit of its computation demand from
    its own users, and $c_i$, which is the cost for one unit of its computation
    consumption, $\forall i \in \mathcal{N}$.
Output: $\mathbf{p}, \mathbf{x}$;
1: All the ESOs send the revenues for one unit of computation demand from
   its own users, and the cost for one unit of its computation consumption to
   the CSO;
2: Having received $r_i$ and $c_i$ from ESO $i$, the CSO computes the optimal pay-
   ment $p_i$ for ESO $i$, $\forall i \in \mathcal{N}$;
3: Upon receiving the payment, each ESO computes its optimal offloaded com-
   putation based on Eq. (15), and sends the results to the CSO;
4: CSO allocates the computation based on $x_i, \forall i \in \mathcal{N}$.
```

to all the ESOs. Each ESO calculates its optimal offloaded computation x_i based on Eq. (15) for the given $\hat{p}_i, \check{p}_i$. Having received $x_i, \forall i \in \mathcal{N}$, the CSO computes $U(\mathbf{x}, \check{\mathbf{p}}), U(\mathbf{x}, \hat{\mathbf{p}})$. If $U(\mathbf{x}, \hat{\mathbf{p}}) < U(\mathbf{x}, \check{\mathbf{p}})$, $\overline{p}_i = \check{p}_i, \forall i \in \mathcal{N}$, otherwise, $\underline{p}_i = \hat{p}_i, \forall i \in \mathcal{N}$. From Line 12 to Line 16 in Algorithm 2, we can see that at each iteration, $\overline{p}_i, \forall i \in \mathcal{N}$ becomes smaller, and $\underline{p}_i, \forall i \in \mathcal{N}$ becomes larger. According to Line 20, $\overline{p} - \underline{p}$ converges a small positive constant. ϵ is set to a small positive constant to make the algorithm more accurate. The above process is repeated until $\overline{p} - \underline{p} < \epsilon$.

Theorem 3. *The proposed Algorithm 2 can reach unique Nash equilibrium.*

Proof. Since U is strict convexity, Algorithm 2 is guaranteed to find $\mathbf{p}^\star$ that maximizeds U. Then, each ESO i can determine its strategy $x_i^\star$ to maximize its utility. Accordingly, the CSO also obtains a fixed utility with $\mathbf{p}^\star$. Hence, the multi-round stackelberg game can reach the unique Nash equilibrium. □

Algorithm 2 Multi-round Stackelberg Game

```
Input: $r_i$, which is the revenue for one unit of its computation demand from
    its own users, and $c_i$, which is the cost for one unit of its computation
    consumption, $\forall i \in \mathcal{N}$.
Output: $\mathbf{p}, \mathbf{x}$;
 1: $\hat{p}_i = \frac{\underline{p}_i + \overline{p}_i}{2}, \check{p}_i = \frac{\hat{p}_i + \overline{p}_i}{2}, \forall i \in \mathcal{N}$;
 2: $\overline{p} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \overline{p}_i, \underline{p} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \underline{p}_i$
 3: while $\overline{p} - \underline{p} \geq \epsilon$ do
 4:   for all $i \in \mathcal{N}$ do
 5:     Calculate $x_i(\hat{p}_i)$ according to Eq. (15);
 6:   end for
 7:   Calculate $U(\mathbf{x}, \hat{\mathbf{p}})$ according to Eq. (4);
 8:   for all $i \in \mathcal{N}$ do
 9:     Calculate $x_i(\check{p}_i)$ according to Eq. (15);
10:   end for
11:   Calculate $U(\mathbf{x}, \check{\mathbf{p}})$ according to Eq. (4);
12:   if $U(\mathbf{x}, \hat{\mathbf{p}}) < U(\mathbf{x}, \check{\mathbf{p}})$ then
13:     $\overline{p}_i = \check{p}_i, \forall i \in \mathcal{N}$;
14:   else
15:     $\underline{p}_i = \hat{p}_i, \forall i \in \mathcal{N}$;
16:   end if
17:   $\hat{p}_i = \frac{\underline{p}_i + \overline{p}_i}{2}, \check{p}_i = \frac{\hat{p}_i + \overline{p}_i}{2}, \forall i \in \mathcal{N}$;
18: end while
19: $p_i = \hat{p}_i, \forall i \in \mathcal{N}$;
20: $\overline{p} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \overline{p}_i, \underline{p} = \frac{1}{|\mathcal{N}|} \sum_{i \in \mathcal{N}} \underline{p}_i$;
21: for all $i \in \mathcal{N}$ do
22:   Calculate $x_i(p_i)$ according to Eq. (15);
23: end for
24: Calculate $U(\mathbf{x}, \mathbf{p})$ according to Eq. (4);
```

6. Dynamics of Computation Offloading

In this section, we consider how dynamical ESOs affect computation offloading. This is natural in practice because the ESOs may leave or join the computation offloading based on the availability of computation in ESOs.

The original Nash equilibrium is denoted by $(\mathbf{x}^\star, \mathbf{p}^\star)$, where $\mathbf{x}^\star$ is the optimal computation for the ESOs at the Nash equlibrium. The new Nash equilibrium after ESOs leaving or joining the computation offloading is denoted by $(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star)$.

6.1. ESO Leaving Computation Offloading

When ESO i leaves computation offloading, $U(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star)$ denotes the utility of CSO at the new Nash equilibrium, $U(\mathbf{x}^\star, \mathbf{p}^\star)$ denotes the utility of CSO at the original Nash equilibrium, and $U(x_i, p_i)$ denotes ESO i's contribution to the CSO's utility at the original Nash equilibrium.

Theorem 4. *When ESO i leaves computation offloading, the new Nash equilibrium satisfies that*

$$U(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star) \geq U(\mathbf{x}^\star, \mathbf{p}^\star) - U(x_i, p_i), \tag{33}$$

Proof. As proved in Sec. 4, $U(\mathbf{x}, \mathbf{p})$ is strictly concave and the optimal strategy $\mathbf{p}^\star$ is guaranteed to be unique, $U(\mathbf{x}, \mathbf{p})$ shows the diminishing return on the payment to the ESOs. In other words, with the increasing number of ESOs, the marginal utility of CSO decreases. $\square$

6.2. ESO Joining Computation Offloading

When an ESO joins the computation offloading, the Nash equilibrium changes only when

$$U(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star) > U(\mathbf{x}^\star, \mathbf{p}^\star). \tag{34}$$

In contrast to ESO leaving, Eq. (34) does not always hold. For example, according to Lemma 5, if $p_i \leq \underline{p_i}$, $\frac{\partial \mathbf{U}}{\partial p_i} = 0$, thus $U(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star) = U(\mathbf{x}^\star, \mathbf{p}^\star)$, the new ESO will be rejected, the Nash equilibrium will be at the same point; if $p_i \geq \overline{p_i}$,

$\frac{\partial \mathbf{U}}{\partial p_i} = -X_i$, thus $U(\bar{\mathbf{x}}^\star, \bar{\mathbf{p}}^\star) < U(\mathbf{x}^\star, \mathbf{p}^\star)$, the ESO will not be joined, and the Nash equilibrium will be sustained.

According to Theorem 3, the new Nash equilibrium is unique. The CSO recomputes the payment of computation by taking the new ESO's offloaded computation into consideration, and broadcasts the calculated payment to all the $|\mathcal{N}| + 1$ ESOs.

7. Simulation Results

In this section, we present simulation results to demonstrate the efficiency of our proposed incentive mechanism. We set the number of ESOs to 5. The total computation of each ESO is 1. The computation demands of ESOs are different, and are given by $1, 0.8, 0.6, 0.4, 0.2$, respectively. We also set the modeling parameter $\alpha = 0.05$. The priority parameter β is chosen as 1. We set the unoffloaded computation for CSO $X_0 = 60$.

7.1. CSO's Utility

Fig. 6 shows the utility of CSO by varing the value of revenue for one unit of ESO's computation demand from its own users. With the increase of unit revenue, ESOs perfer employing the computation for their own users. The amount of computation offloaded is less. As a result, the utility of CSO decreases. Therefore, the proposed computation allocation strategy can provide differentiated computation to the ESOs, and thus efficiently stimulate ESOs to make different contributions to computation offloading.

Fig. 7 illustrates the impact of cost for one unit of ESO's computation consumption. As shown in Fig. 7, with the same revenue, the ESOs with larger unit cost are assigned more computation by the CSO under our computation allocation strategy. This is because a larger unit cost indicates more rewards that can be obtained if the ESO offloads computation to the CSO, or in other words, stronger incentive to stimulate the collaboration between the CSO and the ESOs.

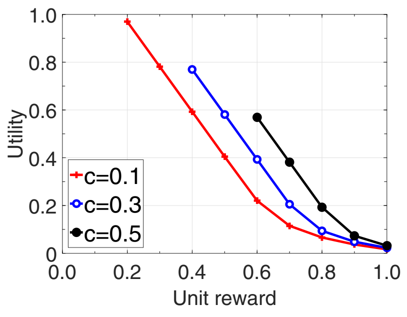


Figure 6: CSO's Utility with different unit rewards.

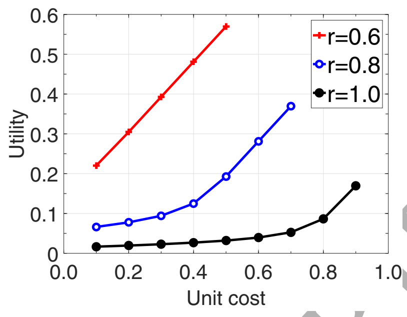


Figure 7: CSO's Utility with different unit costs.

410 ### *7.2. ESO's Utility*

Fig. 8 illustrates the average utility of ESOs by varing the value of unit revenue. As can be seen from Fig. 8, with the increase of unit revenue, the average utility of ESOs increases. The high average utility of ESOs is attributed to the fact that an ESO is willing to serve its own users, leading to highly efficient computation offloading. It is interesting to observe the ESOs achieves a less average utility when the revenue is sufficiently large. This is anti-intuitive, because the ESOs are profit-driven to serve their computation for their own users, and thus, the utility should be the highest. However, large unit revenue can be a double edged sword. While it makes the ESOs willing to serve their own users, it may cause less computation offloaded to the CSO. According to Eq. (13), the utility gain from computation offloading decreases.

We notice from Fig. 9 that although the unit cost increases, the average utility of ESOs increases, serving as the evidence that the ESOs allocate more computation to the their own users. When the unit cost is large enough, on the other hand, the computation used for their own users have reached their demands. As a result, the average utility decreases with even larger unit cost.

β determines the offloading priorities of ESOs. In order to make fair comparison, we set the computation demand of each ESO to 1. The priorities for ESO $1, 2, 3, 4, 5$ are given by $5, 4, 3, 2, 1$, respectively. As illustrated in Fig. 10, the ESO with larger β has higher computation demand for its own mobile users, thus the computation offloaded to the CSO is less, resulting in less utility according to Eq. (13).

7.3. Offloading Ratio

We compare the CSO's computation offloading policy for ESOs with different values of unit revenue and unit cost. To evaluate how the unit revenue impacts the computation offloading ratio, we gradually increases the value from 0.2 to 1.0. The result is shown in Fig. 11. As can be seen, the offloading ratio decreases with the increase of unit revenue. This is because with a higher unit revenue, the ESOs gain more rewards by offloading the computation to their own users, leading to stronger incentive to stimulate the cooperation with its own users.

Fig. 12 illustrates the performance trend by varing the value of unit cost of ESOs. With the increase of unit cost of ESOs, the offloading ratio of computation increases (see Fig. 12). Larger unit cost gains significantly, because it makes ESOs reluctant to serve for their own users and incentive to offload computation to the CSO, thus increasing the offloading ratio.

7.4. ESO Joining

We assume the arrival time of ESOs are $t = 1, 2, 3, 4$ and 5 hour, respectively. The results are obtained using Algorithm 2. We set ϵ to 0.001. We can see from Fig. 13 that the equilibrium between the CSO and ESOs changes when a new ESO joins the computation offloading. The amount of computation for the existing ESOs decrease because a new ESO joins the computation offloading. In addition, at each equilibrium, the computation assigned by the CSO is proportional to the contributions of different ESOs.

7.5. ESO Leaving

We consider the scenario where ESOs leave computation offloading one by one. We assume the same arrival time of ESOs, i.e., 1 hour. The leave time of ESOs 5, 4, 3, 2 are $t = 2, 3, 4, 5$ hour, respectively. From Fig. 14, we can observe that the equilibrium between the CSO and ESOs changes when an ESO leaves

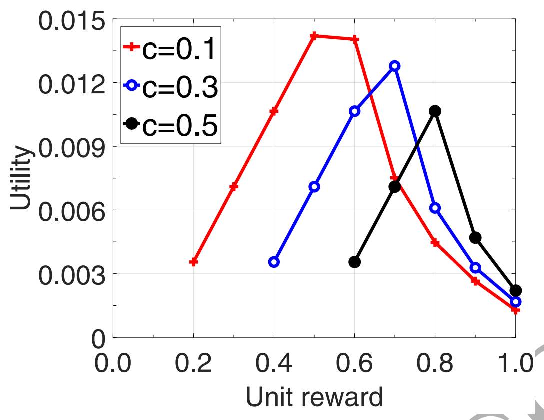


Figure 8: ESO's Utility with different unit rewards.

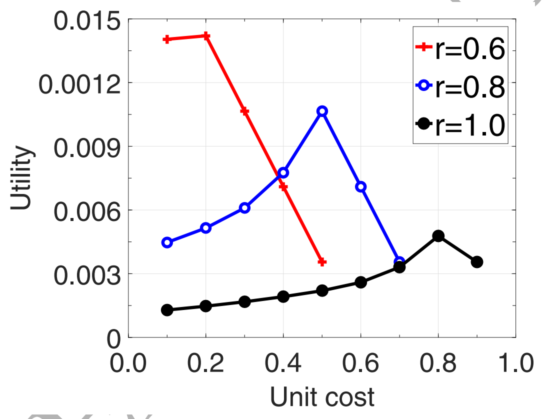


Figure 9: ESO's Utility with different unit costs.

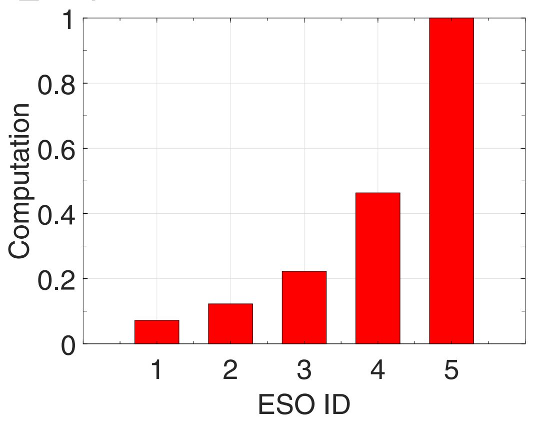


Figure 10: Computation Distribution.

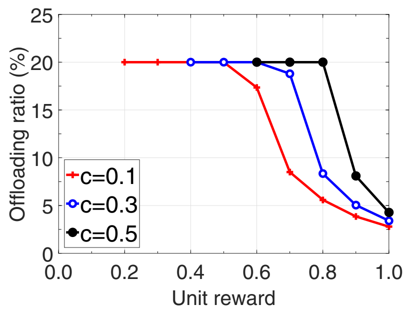


Figure 11: Offloading Ratio with different unit rewards.

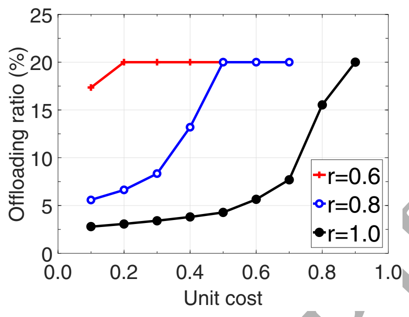


Figure 12: Offloading Ratio with different unit costs.

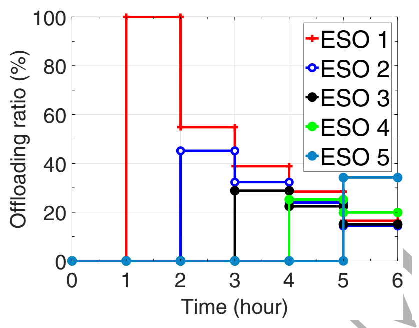


Figure 13: ESO Joining.

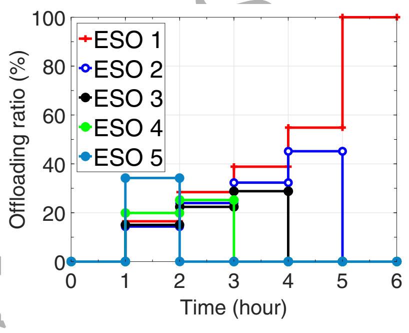


Figure 14: ESO Leaving.

the computation offloading. The computation assigned to the existing ESOs 460 increase due to the decreased number of ESOs. The computation assignment is proportional to the contribution of each ESO at each equilibrium. Note that the equilibrium of this case are the same as those in Sec. 7.4.

7.6. Scalability

Fig. 15 and 16 depict the impact of the number of ESOs. To make fair 465 comparison, we set the computation demand of each ESO to 1. Fig. 15 shows that, with the increase of the number of ESOs, the utility of CSO increases. This is because the increasing number of ESOs fosters the computation offloading to the CSO, thus leading to a larger utility. On the other hand, the average utility of ESOs increases with the increasing number of ESOs at the first beginning,

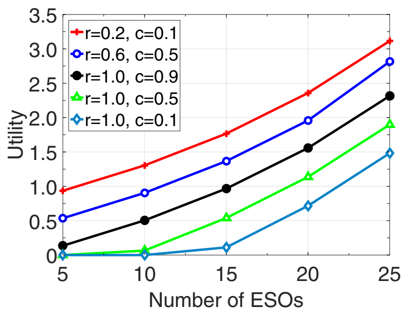


Figure 15: CSO's Utility.

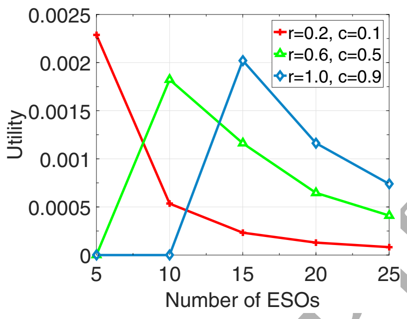


Figure 16: ESO's Utility.

then it decreases if the number of ESOs continues increasing, it is because the computation allocated to each ESO decreases.

8. Conclusion

In this paper, we studied the problem of computation offloading by using edge computing. We formulate the problem as a two-stage stackelberg game and shows it achieves a Nash equilibrium. We design two computation offloading algorithms that have their own advantages of low delay and reduced complexity, respectively. We also consider more general cases which edge server owners may join and leave dynamically. Simulation results corroborate the theoretical analysis as well as validate the effectiveness of the proposed algorithms.

Acknowledgments

This work was supported in part by China Postdoctoral Science Foundation 2015M580051 and 2016T90039, and the National Natural Science Foundation of China under Grant No. 61602038, 61522103, and 61501216.

References

[1] L. Sanchez, L. Muñoz, J. A. Galache, P. Sotres, J. R. Santana, V. Gutierrez, R. Ramdhany, A. Gluhak, S. Krco, E. Theodoridis, D. Pfisterer, Smart-

Santander: IoT Experimentation over a Smart City Testbed, Computer Networks 61 (2014) 217–238.

[2] H. Wu, W. Sun, B. Zheng, Is Only One Gps Position Sufficient to Locate You to the Road Network Accurately?, in: Proc. of ACM UbiComp, 2016, pp. 740–751.

[3] Y. Liu, Y. Han, Z. Yang, H. Wu, Efficient Data Query in Intermittently-Connected Mobile Ad Hoc Social Networks, IEEE Transactions on Parallel and Distributed Systems 26 (5) (2015) 1301–1312.

[4] P.-H. Wu, C.-W. Huang, J.-N. Hwang, J.-Y. Pyun, J. Zhang, Video-Quality-Driven Resource Allocation for Real-Time Surveillance Video Uplinking Over OFDMA-Based Wireless Networks, IEEE Transactions on Vehicular Technology 64 (7) (2015) 3233–3246.

[5] A. Sanjab, W. Saad, Data Injection Attacks on Smart Grids With Multiple Adversaries: A Game-Theoretic Perspective, IEEE Transactions on Smart Grid 7 (4) (2016) 2038–2049.

[6] N. Kumar, S. Misra, J. Rodrigues, J.-H. Lee, M. S. Obaidat, N. Chilamkurti, Playing the Smart Grid Game: Performance Analysis of Intelligent Energy Harvesting and Traffic Flow Forecasting for Plug-In Electric Vehicles, IEEE Vehicular Technology Magazine 10 (4) (2015) 81–92.

[7] N. Kumar, S. Misra, N. Chilamkurti, J.-H. Lee, J. J. P. C. Rodrigues, Bayesian Coalition Negotiation Game as a Utility for Secure Energy Management in a Vehicles-to-Grid Environment, IEEE Transactions on Dependable and Secure Computing 13 (1) (2016) 133–145.

[8] M. F. Bulut, M. Demirbas, H. Ferhatosmanoglu, LineKing: Coffee Shop Wait-Time Monitoring Using Smartphones, IEEE Transactions on Mobile Computing 14 (10) (2015) 2045–2058.

[9] S. Mennicken, O. Zihler, F. Juldaschewa, V. Molnar, D. Aggeler, E. M. Huang, Ït's Like Living with a Friendly Stranger̈: Perceptions of Personality Traits in a Smart Home, in: Proc. of UbiComp, 2016, pp. 120–131.

[10] B. M. C. Silva, J. J. P. C. Rodrigues, N. Kumar, M. L. Proença, Jr., G. Han, MobiCoop: An Incentive-Based Cooperation Solution for Mobile Applications, ACM Transactions on Multimedia Computing, Communications, and Applications 12 (4) (2016) 49:1–49:23.

[11] Ericsson Mobility Report. [Online]. Available: https://www.ericsson.com/mobility-report/.

[12] K. Kumar, Y.-H. Lu, Cloud Computing for Mobile Users: Can Offloading Computation Save Energy?, Computer 43 (4) (2010) 51–56.

[13] J. Baliga, R. W. A. Ayre, K. Hinton, R. S. Tucker, Green Cloud Computing: Balancing Energy in Processing, Storage, and Transport, Proceedings of the IEEE 99 (1) (2011) 149–167.

[14] A. Iosup, S. Ostermann, M. N. Yigitbasi, R. Prodan, T. Fahringer, D. Epema, Performance Analysis of Cloud Computing Services for Many-Tasks Scientific Computing, IEEE Transactions on Parallel and Distributed Systems 22 (6) (2011) 931–945.

[15] S. Yu, C. Wang, K. Ren, W. Lou, Achieving Secure, Scalable, and Fine-grained Data Access Control in Cloud Computing, in: Proc. of IEEE INFOCOM, 2010, pp. 1–9.

[16] S. Kosta, A. Aucinas, P. Hui, R. Mortier, X. Zhang, ThinkAir: Dynamic Resource Allocation and Parallel Execution in the Cloud for Mobile Code Offloading, in: Proc. of IEEE INFOCOM, 2012, pp. 945–953.

[17] L. Yu, H. Shen, K. Sapra, L. Ye, Z. Cai, CoRE: Cooperative End-to-End Traffic Redundancy Elimination for Reducing Cloud Bandwidth Cost, IEEE Transactions on Parallel and Distributed Systems 28 (2) (2017) 446–461.

[18] L. Rao, X. Liu, L. Xie, W. Liu, Minimizing Electricity Cost: Optimization of Distributed Internet Data Centers in a Multi-Electricity-Market Environment, in: Proc. of IEEE INFOCOM, 2010, pp. 1–9.

[19] L. Yu, Z. Cai, Dynamic Scaling of Virtual Clusters with Bandwidth Guarantee in Cloud Datacenters, in: Prof. of INFOCOM, 2016, pp. 1–9.

[20] N. Kumar, R. Iqbal, S. Misra, J. J. P. C. Rodrigues, M. S. Obaidat, Performance-Aware Mobile Game as-a-Service for RFID-Based Secure QoS Management in Mobile Cloud, IEEE Transactions on Emerging Topics in Computing PP (99) (2016) 1–1.

[21] A. Botta, W. De Donato, V. Persico, A. Pescapé, Integration of Cloud Computing and Internet of Things: A Survey, Future Generation Computer Systems 56 (2016) 684–700.

[22] W. Shi, J. Cao, Q. Zhang, Y. Li, L. Xu, Edge Computing: Vision and Challenges, IEEE Internet of Things Journal 3 (5) (2016) 637–646.

[23] C. Xu, S. Jia, L. Zhong, G.-M. Muntean, Socially Aware Mobile Peer-to-Peer Communications for Community Multimedia Streaming Services, IEEE Communications Magazine 53 (10) (2015) 150–156.

[24] C. Xu, S. Jia, M. Wang, L. Zhong, H. Zhang, G.-M. Muntean, Performance-Aware Mobile Community-Based VoD Streaming Over Vehicular Ad Hoc Networks, IEEE Transactions on Vehicular Technology 64 (3) (2015) 1201–1217.

[25] C. Xu, S. Jia, L. Zhong, H. Zhang, G.-M. Muntean, Ant-Inspired Mini-Community-Based Solution for Video-On-Demand Services in Wireless Mobile Networks, IEEE Transactions on Broadcasting 60 (2) (2014) 322–33.

[26] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, A. Patti, CloneCloud: Elastic Execution Between Mobile Device and Cloud, in: Proceedings of the Sixth Conference on Computer Systems, 2011, pp. 301–314.

[27] E. Cuervo, A. Balasubramanian, D.-k. Cho, A. Wolman, S. Saroiu, R. Chandra, P. Bahl, MAUI: Making Smartphones Last Longer with Code Offload, in: Proc. of MobiSys, 2010, pp. 49–62.

[28] W. Chen, Decentralized Computation Offloading Game for Mobile Cloud Computing, IEEE Transactions on Parallel and Distributed Systems 26 (4) (2015) 974–983.

[29] L. Yang, J. Cao, H. Cheng, Y. Ji, Multi-User Computation Partitioning for Latency Sensitive Mobile Cloud Applications, IEEE Transactions on Computers 64 (8) (2015) 2253–2266.

[30] Y. Mao, J. Zhang, K. B. Letaief, Dynamic Computation Offloading for Mobile-Edge Computing with Energy Harvesting Devices, IEEE Journal on Selected Areas in Communications PP (99) (2016) 1–1.

[31] X. Chen, L. Jiao, W. Li, X. Fu, Efficient Multi-User Computation Offloading for Mobile-Edge Cloud Computing, IEEE/ACM Transactions on Networking 24 (5) (2016) 2795–2808.

[32] S. Guo, B. Xiao, Y. Yang, Y. Yang, Energy-Efficient Dynamic Offloading and Resource Scheduling in Mobile Cloud Computing, in: Prod. of INFOCOM, 2016, pp. 1–9.

[33] S. Yang, D. Kwon, H. Yi, Y. Cho, Y. Kwon, Y. Paek, Techniques to Minimize State Transfer Costs for Dynamic Execution Offloading in Mobile Cloud Computing, IEEE Transactions on Mobile Computing 13 (11) (2014) 2648–2660.

[34] R. Kaewpuang, D. Niyato, P. Wang, E. Hossain, A Framework for Cooperative Resource Management in Mobile Cloud Computing, IEEE Journal on Selected Areas in Communications 31 (12) (2013) 2685–2700.

[35] F. Bonomi, R. Milito, J. Zhu, S. Addepalli, Fog Computing and Its Role in the Internet of Things, in: Proceedings of the First Edition of the MCC Workshop on Mobile Cloud Computing, 2012, pp. 13–16.

[36] D. Sabella, A. Vaillant, P. Kuure, U. Rauschenbach, F. Giust, Mobile-Edge Computing Architecture: The Role of MEC in the Internet of Things, IEEE Consumer Electronics Magazine 5 (4) (2016) 84–91.

[37] M. Satyanarayanan, P. Bahl, R. Caceres, N. Davies, The Case for VM-Based Cloudlets in Mobile Computing, IEEE Pervasive Computing 8 (4) (2009) 14–23.

[38] N. Kumar, S. Zeadally, J. J. P. C. Rodrigues, Vehicular Delay-Tolerant Networks for Smart Grid Data Management Using Mobile Edge Computing, IEEE Communications Magazine 4 (10) (2016) 60–66.

[39] X. Hou, Y. Li, M. Chen, D. Wu, D. Jin, S. Chen, Vehicular Fog Computing: A Viewpoint of Vehicles as the Infrastructures, IEEE Transactions on Vehicular Technology 65 (6) (2016) 3860–3873.

[40] F. Jalali, K. Hinton, R. Ayre, T. Alpcan, R. S. Tucker, Fog Computing May Help to Save Energy in Cloud Computing, IEEE Journal on Selected Areas in Communications 34 (5) (2016) 1728–1739.

[41] D. Fudenberg, J. Tirole, Game Theory, MIT Press, 1993.

[42] Q. Wang, W. Wang, S. Jin, H. Zhu, N. T. Zhang, Quality-Optimized Joint Source Selection and Power Control for Wireless Multimedia D2D Communication Using Stackelberg Game, IEEE Transactions on Vehicular Technology 64 (8) (2015) 3755–3769.

[43] A. Belgana, B. P. Rimal, M. Maier, Open Energy Market Strategies in Microgrids: A Stackelberg Game Approach Based on a Hybrid Multiobjective Evolutionary Algorithm, IEEE Transactions on Smart Grid 6 (3) (2015) 1243–1252.

[44] X. Kang, Y. Wu, Incentive Mechanism Design for Heterogeneous Peer-to-Peer Networks: A Stackelberg Game Approach, IEEE Transactions on Mobile Computing 14 (5) (2015) 1018–1030.

[45] H. Yu, M. H. Cheung, L. Gao, J. Huang, Economics of Public Wi-Fi Monetization and Advertising, in: Proc. of IEEE INFOCOM, 2016, pp. 1–9.

[46] H. Yang, X. Xie, A. V. Vasilakos, Noncooperative and Cooperative Optimization of Electric Vehicle Charging Under Demand Uncertainty: A Robust Stackelberg Game, IEEE Transactions on Vehicular Technology 65 (3) (2016) 1043–1058.

[47] J. F. Nash, Non-Cooperative Games, The Annals of Mathematics 54 (2) (1951) 286–295.

[48] Y. Liu, Z. Yang, T. Ning, H. Wu, Efficient Quality-of-Service (QoS) Support in Mobile Opportunistic Networks, IEEE Transactions on Vehicular Technology 63 (9) (2014) 4574–4584.

[49] Y. Liu, H. Wu, Y. Xia, Y. Wang, F. Li, P. Yang, Optimal Online Data Dissemination for Resource Constrained Mobile Opportunistic Networks, IEEE Transactions on Vehicular Technology PP (99) (2016) 1–1.

Biography

Yang Liu received the BE degree in electrical engineering and its automation and the ME degree in control theory and control engineering from Harbin Engineering University, Harbin, China, in 2008 and 2010, respectively, and the PhD degree in computer engineering at the Center for Advanced Computer Studies, University of Louisiana at Lafayette, Lafayette, in 2014. He is currently an assistant professor at Beijing University of Posts and Telecommunications. His current research interests include wireless networking and mobile computing.

Changqiao Xu received the Ph.D. degree from Institute of Software, Chinese Academy of Sciences (ISCAS), Beijing, China, in 2009. He was an Assistant Research Fellow with ISCAS from 2002 to 2007. From 2007 to 2009, he was a Researcher with the Software Research Institute, Athlone Institute of Technology, Athlone, Ireland. He joined Beijing University of Posts and Telecommunications, Beijing, in 2009. He is currently a Professor with State Key Laboratory of Networking and Switching Technology, and the Director of the Next Generation Internet Technology Research Center, BUPT. He has authored over 100 technical papers in prestigious international journals and conferences. His research interests include wireless networking, multimedia communications, and next generation Internet technology. Dr. Xu served as the Co- Chair of the IEEE MMTC Interest Group, Green Multimedia Communications, and a Board Member of the IEEE MMTC Services and Publicity.

Yufeng Zhan received his BS degree in computer science from Anhui University of Architecture, Hefei, China, in 2012 and MS degree in Control Engineering from Beijing Institute of Technology, Beijing, China, in 2014, where he is currently working toward his Ph.D. degree in control science and engineering in the School of Automation at Beijing Institute of Technology. His research interests include networked control systems, wireless sensor networks, and mobile opportunistic networks.

Zhixin Liu received his B.S., M.S., and Ph.D. degrees in control theory and engineering from Yanshan University, Qinhuangdao, China, in 2000, 2003, and 2006, respectively. He is currently a professor with the Department of Automation, Institute of Electrical Engineering, Yanshan University, China. He visited the University of Alberta, Edmonton, AB, Canada, in 2009. His current research interests include performance optimization and energy-efficient protocol design in wireless sensor networks, wireless resource allocation in cognitive radio networks and Femtocell networks.

Jianfeng Guan received his B.S. degree from Northeastern University of China in 2004, and the Ph.D. degree in communications and information system from Beijing Jiaotong University in 2010. He is currently an associate professor at the Institute of Network Technology, Beijing University of Posts and Telecommunications. His main research interests focus on future networks, network security, mobility management and routing.

Hongke Zhang received his M.S. and Ph.D. degrees in electrical and communication systems from the University of Electronic Science and Technology of China (UESTC), Chengdu, China, in 1988 and 1992, respectively. From 1992 to 1994, he was a post-doctor at Beijing Jiaotong University (BJTU), Beijing, China. He is currently a professor at the School of Electronic and Information Engineering, BJTU, China and the director of a National Engineering Lab on Next Generation Internet Technologies, China. His research has resulted in many papers, books, patents, systems and equipment in the areas of communications and computer networks. He is the author of more than 10 books and the holder of more than 70 patents. He is the Chief Scientist of a National Basic Research Program of China (973 Program) and has also served on the editorial board of several international journals.