Kim *EURASIP Journal on Wireless Communications and Networking* (2015) 2015:229
DOI 10.1186/s13638-015-0456-5

EURASIP Journal on Wireless Communications and Networking
a SpringerOpen Journal

RESEARCH Open Access

Nested game-based computation offloading scheme for Mobile Cloud IoT systems

Sungwook Kim

Abstract

With an explosive growth of Mobile Cloud and Internet of Things (IoT) technologies, the Mobile Cloud IoT (MCIoT) concept has become a new trend for the future Internet. MCIoT paradigm extends the existing facility of computing process to different mobile applications executing in mobile and portable devices. In this research, we provide a new nested game model to design an effective MCIoT computation offloading algorithm. First, each mobile device determines the portion of remote offloading computation based on the Rubinstein game approach. Then, a computation resource in the cloud system is dynamically assigned for the requested offloading computation. Based on the nested game principle, our proposed scheme can approach an optimal solution for the offloading computation in the MCIoT system. The simulation results show that our game-based method is effective in distributed IoT environments while supporting application executions timely and ubiquitously.

Keywords: Mobile Cloud Computing; Computation offloading; Dynamic resource allocation; Game theory; Nested game; Rubinstein game; Internet of Things

1 Introduction

Recently, rapid technology development makes it possible for connecting various smart mobile devices together while providing more data interoperability methods for application purpose. Therefore, the diverse nature of applications has challenged communication and computation mechanisms to look beyond conventional applications for effective network policies, quality of service (QoS), and system performance. The Internet of Things (IoT) paradigm is based on intelligent and self-configuring mobile devices interconnected in a dynamic and global network infrastructure. It is enabling ubiquitous and pervasive computing scenarios in the real world [1, 2].

In the current decade, a growing number of researches have been conducted to acquire data ubiquitously, process data timely, and distribute data wirelessly in the IoT paradigm. To satisfy this requirement, mobile devices should have the capacity to handle the required processing and computation work. Unfortunately, the desire for rich, powerful applications on mobile devices conflicts with the reality of these devices' limitations: slow computation processors, little memory storage, and limited battery life. For this reason, mobile devices still lag behind desktop and server hardware to provide the experience that users expect [3, 4].

The Mobile Cloud (MC) is emerging as one of the most important branches of cloud computing and is expected to expand mobile ecosystems. Usually, cloud computing has long been recognized as a paradigm for big data storage and analytics; it has virtually unlimited capabilities in terms of storage and processing power. MC is the combination of cloud computing, mobile computing, and wireless networks to bring rich computational resources to mobile users and network operators, as well as cloud computing providers. With the explosive growth of multimedia mobile applications, MC computing has become a significant research topic of the scientific and industrial communities [5–7].

The two fields of MC and IoT have been widely popular as future infrastructures and have seen an independent evolution. However, MC and IoT are complementary technologies, and several mutual advantages deriving from their integration have been identified [3]. Therefore, a symbiosis has developed between mobile devices and MC and is expected to be combined for the future Internet. Generally, mobile devices can benefit from the

Correspondence: swkim01@sogang.ac.kr
Department of Computer Science, Sogang University, 35 Baekbeom-ro (Sinsu-d ong), Mapo-gu, Seoul 121-742, South Korea

virtually unlimited capabilities and resources of MC to compensate its storage, processing, and energy constraints. Specifically, MC can offer an effective solution to implement IoT service management. On the other hand, MC can benefit from the IoT system by extending its scope to deal with real-world things in a more distributed and dynamic manner [2]. Nowadays, the extension of MC over dynamic IoT environments has been referred as a next generation communication and computing paradigm [5].

In this paper, we focus our attention on the integration of MC and IoT, which we call the MCIoT paradigm. MCIoT should support a wide variety of multimedia applications with different QoS requirements; these applications need different system resources. Therefore, the MCIoT platform can benefit from implementing QoS-aware service management algorithms to match application demand to IoT services, while guaranteeing to meet the respective QoS requirements. Such algorithms must take into account that mobile devices are generally characterized with limited storage and processing capacity. A possible way to dealing with this problem is to remotely execute some computation tasks on a more powerful cloud system, with results communicated back to the mobile devices. This method is the computation offloading [8]. Therefore, the synergy of MC and IoT lies at the junction of mobile devices, different wireless network providers, and cloud computing systems. As far as we know, there is no intensive research work considering the multi-interaction aspect of the MCIoT paradigm.

Although the computation offloading approach can significantly augment the computation capability of mobile devices, the task of developing a comprehensive and reliable computation offloading mechanism remains challenging. A key challenge is how to efficiently coordinate multiple mobile devices and the MCIoT system. Usually, individual mobile devices locally make control decisions to maximize their profits in a distributed manner. This situation leads us into the game theory. The game theory is a conceptual and procedural tool to capture the interaction among selfish players and has been successfully applied to typical network management problems, like resource allocation, routing, pricing, and load balancing. In particular, it has been largely applied in mobile and wireless network scenarios in the context of competition in the access to shared communication and computation resources [9].

In 1988, an American political scientist, George Tsebelis, introduced an important new concept, called nested games, to rational choice theory and to the study of comparative politics [10]. Using the notion of nested games, he showed that game players are involved simultaneously in several games. He argued that the "nestedness" of the principal game explains why a player confronted with a series of choices might not pick the alternative which appears to be optimal. In other words, what seems to be irrational in one arena becomes intelligible when the whole network of games is examined. Originally, the nested game has been used by anyone interested in the effects of political context and institutions on the behavior of political actors. Nowadays, the nested game approach can be used to analyze a systematic, empirically accurate, and theoretically coherent account of apparently irrational actions [11–13].

In this paper, we adopt a nested game approach to address the computation offloading algorithm in the MCIoT platform. The nested game model is a useful framework for designing decentralized mechanisms, such that the mobile devices can self-organize into the mutually satisfactory computation offloading decisions. Usually, different mobile devices may pursue different interests and act individually to maximize their profits. This self-organizing feature can add autonomics into MCIoT systems and help to ease the heavy burden of complex centralized control algorithms. The main contributions of our work are (i) the ability to analyze the interactions among multiple mobile devices, (ii) the ability to effectively assign an available resource for the requested offloading computations, and (iii) the ability to respond to the current MCIoT system conditions while maximizing the performance.

A. Related work

In modern times, a lot of state-of-the-art work on cloud-based IoT management schemes has been conducted. In [14], an effective approach to intelligent planning for mobile IoT applications was presented. This approach included a learning technique for dynamically assessing the users' mobile IoT application and a Markov decision process planning technique for enhancing efficiency of IoT device action planning. In [15], a highly localized IoT-based cloud computing model was proposed. This model allowed clients to create ad hoc clouds using the IoT and other computing devices in the nearby physical environment, while providing the flexibility of cloud computing. It also provided localized computation capability from untapped computing resources. In [16], a novel framework for software-defined IoT cloud systems was developed. With feasibility and practical applicability, this framework handled two main tasks: (i) perform dynamic, on-demand provisioning of governance capabilities and (ii) remotely invoke such capabilities in IoT cloud resources remotely, via dynamic APIs.

Recently, related work on an efficient computation offloading mechanism design is reviewed in [11] and [7]. The *nested two-stage game-based optimization* (*NTGO*) scheme [11] was developed for an MC computation

interaction system. This scheme provides a nested two-stage game model for the computation offloading. To maximize the network system performance, all the mobile devices compete for the allocated resources, which becomes a normal-form game. Based on the backward induction principle, the *NTGO* scheme can derive the near-optimal strategy for all the mobile devices using a convex optimization approach. Nash equilibrium always exists and is unique in the *NTGO* scheme. The *decentralized computation offloading game* (*DCOG*) scheme [7] was envisioned as a promising approach for an MCC algorithm. This scheme formulated a decentralized computation offloading mechanism as a decentralized computation offloading game. As game players, mobile devices make computation offloading decisions locally, which can significantly reduce the controlling and signaling overhead of the cloud. The *DCOG* scheme also can achieve a Nash equilibrium of the game.

All of the abovementioned solutions in [7, 11] have attracted a lot of attention and introduced unique challenges to efficiently solve the computation offloading decision problems. However, there are several disadvantages. First, these existing schemes rely on high complexity and extra overhead; this increased overhead can exhaust the computation capacity. Second, these schemes cannot adaptively estimate the current system conditions. Third, these schemes operate systems by fixed-system parameters. Under dynamic MCIoT environments, control mechanisms by using static thresholds can cause potential erroneous decisions. Compared to these schemes [7, 11] in Section III, the proposed scheme attains a better system performance.

This paper is organized as follows. Section II explains the basic concept of the network model for computation offloading. Section III describes our proposed nested game-based computation offloading scheme in detail. In Section IV, performance evaluation results are presented along with comparisons with the schemes proposed in [7, 11]. Through simulation, we show the ability of the proposed scheme to achieve high accuracy and promptness in dynamic network environments. Finally, concluding remarks are given in Section V.

2 Network model for computation offloading

Under competitive or cooperative environments, the behaviors of game players have a direct influence on each other. Based on rational assumptions, the game theory is to study the decision-making mechanism and the balance of decision-making interactions [9]. However, game players seem to act irrationally, but if treating the game as a part of a larger game, we can see their behaviors are rational [12]. From the view of a small independent game, each player's strategy is not the optimal solution. However, from the view of a big game, players' reactions are the best responses. Such games are called as nested games, and small games may be used as sub-games nested in the sequential game of a larger game [11–13]. In multiple fields of a nested game, game players try to optimize their payoffs in the principal game field which also involves a game about the rules of the game. It can lead to apparently suboptimal payoffs as the game player fails to see the other fields that provide context for the small game in the principal field. Several studies use nested game theory to explain political party behavior, budget negotiations, electoral systems, and public policy [13].

In this work, we specify the nested game to design a new computation offloading algorithm. The key point of the MC offloading mechanism hinges on the ability to achieve enough computing resources with small energy consumption. In recent years, this technique has received more attention because of the significant rise of offload-available mobile applications, the availability of powerful clouds and the improved connectivity options for mobile devices. The main challenges to design an effective computation offloading algorithm lies in the adaptive division of applications for partial offloading, the mismatch control mechanism between how individual mobile devices demand and access computing resources, and how cloud providers offer them. To decide what, when, and how to be offloaded, we should consider the offload overhead and current MCIoT system conditions.

In the resource-rich MC environment, a mobile device must pay the price to take advantage of computation offloading. To gain an extra benefit, idle computation resources in the MCIoT system compete to get the requested offloading task. Therefore, mobile devices can select the most adaptable computation resource to execute their offloaded computations. In this work, our MCIoT environment can be described as follows:

1. $\mathbb{D} = \{D_0, D_1, \ldots, D_n\}$ is the set of mobile devices, and A_i, $i \in [1, \ n]$ is an application, which belongs to the mobile device D_i.
2. Mobile device applications are elastic applications and can be split. For example, $A_i = \sum_{k=1}^{L} a_k^i$, where a_k^i is the kth module of A_i, and some parts (i.e., coded modules) of A_i can be offloaded.
3. $\mathbb{R} = \{R_0, R_1, \ldots, R_m\}$ is the set of idle cloud computing resources in the MCIoT environment, and R_j, $j \in [1, \ m]$ has a computation capacity ($\mathcal{F}_{R_j}^R$; CPU cycles per second) and expected price (ψ^{R_j}; price per CPU cycle of the R_j) to accomplish the offloaded computations.
4. Price in each $R_{j,1 \le j \le m}$ can be dynamically adjustable according to the auction mechanism.
5. For simplicity, we assume that there is no communication noise and uncertainties in MCIoT environments.

In this paper, we develop a two-stage nested game model comprised of elastic applications, mobile devices, and computation resources in the MC. In the first stage, applications of mobile devices are divided into two parts: one part runs locally and the other part is run on the MC side. In the second stage, offloaded tasks are matched to computing resources in the MCIoT system. Based on the auction mechanism, computation resources submit different offers to get the requested offload task, and the most adaptable offer is selected. According to the two-stage sequential nested game approach, we can make decisions about whether to perform computation offloading, which portion of the application should be offloaded to the cloud, and which resource is selected to accomplish the requested offload.

3 Proposed computation offloading algorithms

In this section, we introduce our proposed computation offloading scheme in detail. The computation offloading technique has been widely popular as the future infrastructure of utility computing and becomes an attractive tool to overcome the inherent limitations of mobile devices. Based on the nested game model, our proposed scheme consists of the Rubinstein game and auction game to approximate a globally desirable system performance while ensuring QoS for mobile application services.

A. Offloading communication and computation process

By taking into account both communication and computation aspects of MCIoT environments, we formulate a new decentralized computation offloading algorithm. In each mobile device, applications can be computed either locally on the mobile device or remotely on the cloud via computation offloading. For the local computing approach, each mobile device (e.g., D_i) can execute some computation part of A_i individually. The local computation execution time ($L\_CT_{\text{comp}}^{A_i\_D_i}$) of the application A_i on the mobile device D_i is given as

$$L\_CT_{\text{comp}}^{A_i\_D_i} = \frac{\sum_{k=1}^{L} \mathfrak{U}\left(a_k^i\right) \times a_k^i}{\mathcal{F}_{D_i}^{L}},\ \text{s.t.},\ \mathfrak{U}\left(a_k^i\right) = \begin{cases} 1, & \text{if } a_k^i \text{ is locally computed} \\ 0, & \text{otherwise } \left(a_k^i \text{ is offloaded}\right) \end{cases} \tag{1}$$

where $\mathcal{F}_{D_i}^{L}$ is the computation capability of mobile device D_i. The local computation cost ($L\_CC_{\text{comp}}^{A_i\_D_i}$) of the application A_i on the mobile device D_i is calculated based on the $L\_CT_{\text{comp}}^{A_i\_D_i}$ and consumed local computation energy (ρ).

$$L\_CC_{\text{comp}}^{A_i\_D_i} = \rho^{D_i} \times \left(\mathcal{F}_{D_i}^{L} \times L\_CT_{\text{comp}}^{A_i\_D_i}\right) \tag{2}$$

where ρ^{D_i} is the coefficient denoting the consumed energy cost per CPU cycle. The evaluation of the total local computation overhead ($T\_O_{\text{local}}^{A_i\_D_i}$) of the application A_i on the mobile device D_i is a non-trivial multi-objective optimization problem. It is addressed as a weighted sum by considering normalized time and energy cost.

$$T\_O_{\text{local}}^{A_i\_D_i} = \lambda_{D_i}^{A_i} \times \left(\frac{L\_CT_{\text{comp}}^{A_i\_D_i}}{\frac{1}{\mathcal{F}_{D_i}^{L}} \times \sum_{k=1}^{L} a_k^i}\right) + \left(1-\lambda_{D_i}^{A_i}\right) \times \left(\frac{L\_CC_{\text{comp}}^{A_i\_D_i}}{\rho^{D_i} \times \sum_{k=1}^{L} a_k^i}\right) \tag{3}$$

where $\lambda_{D_i}^{A_i}$ is a parameter to control the relative weights given to execution time and energy consumption. To satisfy A_i's demand, $\lambda_{D_i}^{A_i}$ is adaptively decided. In Eq. (7), the $\lambda_{D_i}^{A_i}$ value decision process is explained in detail.

Next, we estimate the remote computation overhead through the MC offloading mechanism. Generally, the communication and computation aspects play a key role in MC offload. In this paper, we consider a delay-sensitive Wi-Fi model for offloading services; mobile devices are sensitive to delay and their payoff decreases as delay increases. As a wireless access base station (BS), a Wi-Fi access point manages the uplink/downlink communications of mobile devices. For the computation offloading, the mobile device D_i would incur the extra overhead in terms of time and energy to submit the computation offload via wireless access. Based on the communication model in [7], the offloading communication time ($O\_CT_{\text{off}}^{A_i\_D_i}$) and energy ($O\_CE_{\text{off}}^{A_i\_D_i}$) of the application A_i on the mobile device D_i are computed as follows.

$$O\_CT_{\text{off}}^{A_i\_D_i}(\mathbb{D}) = \frac{\sum_{k=1}^{L} \mathfrak{T}\left(a_k^i\right) \times a_k^i}{\mathscr{B} \times \log_2\left(1 + \frac{P_i \times H_{i,\,\text{BS}}}{\omega + \sum_{a_k^g \in A_g \setminus \{A_i\}: \mathfrak{T}\left(a_k^g\right)=1} P_g \times H_{g,\,\text{BS}}}\right)}$$
$$s.t.,\ g \in [1,\ n], D_g \in D \text{and} \ \mathfrak{T}\left(a_k^i\right) = \begin{cases} 1, & \text{if } a_k^i \text{ is offloaded} \\ 0, & \text{otherwise} \end{cases}$$
$$O\_CE_{\text{off}}^{A_i\_D_i}(\mathbb{D}) = \frac{P_i \times \sum_{k=1}^{L} \mathfrak{T}\left(a_k^i\right) \times a_k^i}{\mathscr{B} \times \log_2\left(1 + \frac{P_i \times H_{i,\text{BS}}}{\omega + \sum_{a_k^g \in A_g \setminus \{A_i\}: \mathfrak{T}\left(a_k^g\right)=1} P_g \times H_{g,\,\text{BS}}}\right)} \tag{4}$$

where $\mathscr{B}$ is the channel bandwidth and P_i is the transmission power of device D_i. $H_{i,\,\text{BS}}$ denotes the channel gain between the mobile device D_i and the BS, and ω denotes the interference power [7]. From the Eq. (4), we can see that if too many mobile devices choose to offload the computation via wireless access simultaneously,

they may incur severe interference, leading to low data rates. It would negatively affect the performance of MC communication. Therefore, offloading decisions among mobile devices are tightly coupled to each other [7]. To address this conflicting situation, game theory can be adopted to achieve efficient computation offloading decisions.

After the offloading, the computation time $\left(C\_T_{\text{remote}}^{A_i\_R_j}\right)$ and payment $\left(P_{\text{remote}}^{A_i\_R_j}\right)$ of remote computation task $\left(\sum_{k=1}^{L}\mathfrak{T}\left(a_k^i\right)\times a_k^i\right)$ on the assigned computation resource R_j can be then given as

$$C\_T_{\text{remote}}^{A_i\_R_j}=\frac{\sum_{k=1}^{L}\mathfrak{T}\left(a_k^i\right)\times a_k^i}{\mathcal{F}_{R_j}^{R}}\quad\text{and}\quad P_{\text{remote}}^{A_i\_R_j}=\psi^{R_j}\times\mathcal{F}_{R_j}^{R}\times C\_T_{\text{remote}}^{A_i\_R_j}\tag{5}$$

where $\mathcal{F}_{R_j}^{R}$ is R_j's computation capability and ψ^{R_j} is the coefficient denoting the price per CPU cycle of R_j. According to Eqs. (4) and (5), the total offload overhead $\left(T\_O_{\text{off}}^{A_i\_R_j}\right)$ of application A_i on the computation resource R_j is computed as a weighted sum by considering execution time and consuming cost.

$$\begin{aligned}T\_O_{\text{off}}^{A_i\_R_j}=\lambda_{D_i}^{A_i}&\times\left(\frac{O\_CT_{\text{off}}^{A_i\_D_i}(\mathbb{D})+C\_T_{\text{remote}}^{A_i\_R_j}}{\frac{1}{\mathcal{F}_{D_i}^{L}}\times\sum_{k=1}^{L}a_k^i}\right)\\&+\left(1-\lambda_{D_i}^{A_i}\right)\\&\times\left(\frac{O\_CE_{\text{off}}^{A_i\_D_i}(\mathbb{D})+P_{\text{remote}}^{A_i\_R_j}}{\rho^{D_i}\times\sum_{k=1}^{L}a_k^i}\right)\end{aligned}\tag{6}$$

According to Eqs. (3), (4), (5), and (6), the total execution time of $A_i\left(T\_C_{\text{total}}^{A_i}\_D_i,\ R_j\right)$ with partial offloading can be estimated considering between the local and remote computing times.

$$T\_C_{\text{total}}^{A_i}\_D_i,\ R_j=\mathbf{max}\left[L\_CT_{\text{comp}}^{A_i\_D_i},\ \left(O\_CT_{\text{off}}^{A_i}\_D_i(\mathbb{D})+C\_T_{\text{remote}}^{A_i\_R_j}\right)\right]\tag{7}$$

Finally, we can compute the total execution overhead of $A_i\left(\mathcal{S}^{A_i\_D_i,\ R_j}(\mathbb{D})\right)$ as

$$\begin{aligned}\mathcal{S}^{A_i\_D_i,\ R_j}(\mathbb{D})=\lambda_{D_i}^{A_i}&\times\left(T\_C_{\text{total}}^{A_i}\_D_i,\ R_j\right)\\&+\left(1-\lambda_{D_i}^{A_i}\right)\\&\times\left(\frac{L\_CC_{\text{comp}}^{A_i}\_D_i+\left(O\_CE_{\text{off}}^{A_i\_D_i}(\mathbb{D})+P_{\text{remote}}^{A_i\_R_j}\right)}{\rho^{D_i}\times\sum_{k=1}^{L}a_k^i}\right)\end{aligned}\tag{8}$$

To meet the application-specific demand, different applications have different evaluation criteria for time and energy consumption. For example, when a mobile device is running an application that is delay sensitive (e.g., real-time applications), it should put more weight on the execution time (i.e., a higher λ) in order to ensure the time deadline and vice versa. Therefore, a fixed value for λ cannot effectively adapt to the different application demands. In this work, the value of λ for the A_i on mobile device $D_i\left(\lambda_{D_i}^{A_i}\right)$ is dynamically decided as follows.

$$\lambda_{D_i}^{A_i}=\mathbf{mim}\left[1,\ \left(\frac{\frac{1}{\mathcal{F}_{D_i}^{L}}\times\sum_{k=1}^{L}a_k^i}{T\_D^{A_i}}\right)\right]\tag{9}$$

where $T_{D^{A_i}}$ is the time deadline of A_i. Therefore, through the real-time online monitoring, we can be more responsive to application demands.

B. Application partitioning game

An important challenge for partial offloading is how to partition elastic applications and which part of the partitioned application should be pushed to the remote clouds. In this section, we analyze how mobile devices can exploit a partial offloading between cloud computation and local computation. To distribute computation tasks for partial offloading, we provide a non-cooperative bargaining game model by considering the consuming cost and computation time. Usually, a solution to the bargaining game model enables the game players to fairly and optimally determine their payoffs to make joint agreements [17, 18]. Therefore, the bargaining model is attractive for the partitioning problem.

In 1982, Israeli economist *Ariel Rubinstein* built up an alternating-offer bargain model based on Stahl's limited negotiation model; it is known as a *Rubinstein-Stahl* bargaining process. This model can provide a possible solution to the problem that two players are bargaining with the division of the benefits [19, 20]. In the *Rubinstein-Stahl* model, players have their own bargaining power (δ). The division proportion of the benefits can be obtained according to the bargaining power, which can be

computed at each player individually. Usually, the bargaining solution is strongly dependent on the bargaining powers. If different bargaining powers are used, the player with a higher bargaining power obtains a higher benefit than the other players. In this study, players negotiate with each other by proposing offers alternately. After several rounds of negotiation, players finally reach an agreement as follows [19, 20].

$$(x_1^*, x_2^*) = \begin{cases} \left(\frac{1-\delta_2}{1-\delta_1\delta_2}, \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}\right) & \text{if the } player\_1 \text{ offers first} \\ \left(\frac{\delta_1(1-\delta_2)}{1-\delta_1\delta_2}, \frac{1-\delta_1}{1-\delta_1\delta_2}\right) & \text{if the } player\_2 \text{ offers first} \end{cases} \tag{10}$$

$$\text{s.t.}, (x_1^*, x_2^*) \in R^2 : x_1^* + x_2^* = 1, x_1^* \geq 0, x_2^* \geq 0 \text{ and } 0 \leq \delta_1, \delta_s \leq 1$$

It is obvious that $\frac{1-\delta_2}{1-\delta_1\delta_2} \geq \frac{\delta_2(1-\delta_1)}{1-\delta_1\delta_2}$ and $\frac{\delta_1(1-\delta_2)}{1-\delta_1\delta_2} \leq \frac{1-\delta_1}{1-\delta_1\delta_2}$. Traditionally, the bargaining power in the *Rubinstein-Stahl*'s model is defined as follows [20].

$$\delta = e^{-\xi \times \Phi}, \text{s.t.}, > 0 \tag{11}$$

where Φ is the time period of a negotiation round. Given that Φ is fixed, δ is monotonically decreasing with ξ. Therefore, ξ is an instantaneous discount factor to adaptively adjust the bargaining power. Usually, the bargaining power represents the relative ability to exert influence over other players; the more bargaining power a player has, the more payoff a player attains.

In this section, the *Rubinstein-Stahl* bargaining game model is formulated to solve the application partitioning problem. In our game model, the cloud computation resource and mobile device are assumed as players, which are denoted as *player_1* (i.e., mobile device for local computation) and *player_2* (i.e., cloud resource for remote computation). In the scenario of the *Rubinstein-Stahl* model, each player has a different discount factor (ξ). Under various MCIoT situations, we dynamically adjust ξ values to provide more efficient control over system condition fluctuations. When the current local computation overhead is heavy, the mobile device does not have sufficient computation capacity to support the local computation service. In this case, a higher value of *player_1*'s discount factor (ξ_{I}) is more suitable. If the reverse has been the case (i.e., the remote computation overhead is heavy), a higher value of *player_2*'s discount factor (ξ_{II}) is suitable. At the end of each game period, *player_1* and *player_2* adjust their discount factor values (ξ_{I} and ξ_{II}, respectively) as follows.

$$\xi_{\mathrm{I}} = 1 - \xi_{\mathrm{II}}, \quad \text{s.t.}, \quad \xi_{\mathrm{II}} = \frac{T\_O_{\mathrm{off}}^{A_i - R_j}}{\mathcal{S}^{A_i - D_i,\, R_j}(\mathbb{D})} \tag{12}$$

For simplicity, we assume that the ξ values are fixed within an offloading procedure for each application, while they can be changed in different applications. Therefore, as system situations change after application partitioning, each player can adaptively adjust their ξ_{I} and ξ_{II} values for the next application execution while responding current MCIoT system conditions.

C. Cloud resource selection game

Recently, researchers have proposed various auction models to optimally match up the buyer and seller according to their desires. It is a significant and efficient market-based approach to solve the allocation problem with more requisitions. Therefore, the auction game model can provide a resource-selection mechanism in MC systems. In our computation resource-selection scenario, there are a requested offload task (i.e., buyer) and computation resources (i.e., sellers). Based on the sequential offloading requests, our action model is designed as the one-to-many auction structure. As sellers, computation resources ($\mathbb{R}$) in the MC system offer bids (i.e., the expected selling prices) for the remote offload computation. To show their preference to get the offloading computation, sellers ($\mathbb{R}$) can adjust their selling prices periodically. For the tth auction stage, the seller (i.e., $\boldsymbol{R_j} \in \mathbb{R}$) bids his price $(\boldsymbol{\psi}^{\boldsymbol{R_j}}(t))$ per CPU cycle as follows.

$$\psi^{R_j}(t) = \begin{cases} \psi^{R_j}(t-1) + \left(1 - \frac{1}{(\exp(\max[0, \varepsilon_{R_j}]))}\right), & \text{if } R_j \text{ is selected at } t-1 \\ & s.t., \quad \varepsilon_{R_j} \sim N\left(\mu_{R_j}, \sigma_{R_j}^2\right) \\ \psi^{R_j}(t-1) - \left(1 - \frac{1}{(\exp(\max[0, \varepsilon_{R_j}]))}\right), & \text{otherwise} \end{cases} \tag{13}$$

where $\boldsymbol{\varepsilon_{R_j}}$ is a random variable to present the price adjustment. Because the sellers are not interrelated, the random variable ($\boldsymbol{\varepsilon}$) of each seller is independent of each other. According to Eq. (13), each seller (i.e., $\boldsymbol{R_j} \in \mathbb{R}$) bids his offer $\left(\boldsymbol{\psi}^{\boldsymbol{R_j}}, \ \mathscr{F}_{\boldsymbol{R_j}}^{\boldsymbol{R}}\right)$ at each auction round, and then, the buyer (i.e., $\boldsymbol{D_i} \in \mathbb{D}$) selects the most adaptable offer. In this work, the minimum price-offering resource while satisfying the computation deadline ($\boldsymbol{T\_D}^{(\cdot)}$) is selected. This dynamic auction procedure is repeated sequentially for each auction round. In each auction round, sellers can learn the buyer's desire with incoming information and can make a better price decision for the next auction.

D. The main steps of proposed algorithm

A more recent and rapidly increasing trend deals with two technologies (MC and IoT) together, and they have combined synergistically. Without the MC computing technology, many mobile applications in the IoT system

would not exist. The complementary characteristics of MC and IoT inspire a new MCIoT paradigm. This integration realizes a new convergence scenario and will impact future application development where new opportunities arise for data aggregation, integration, and sharing with third parties. Although the MC and IoT may seem to be a mature technology, this work is only at the very beginning of exploiting the potential approach of the MC and IoT integration.

In this paper, we present a two-stage nested game model for the interaction of game players such as elastic applications, mobile devices, and MC computation resources. Applications are involved in the application partitioning game in the first stage, and mobile devices and MC resources are involved in the resource selection game in the second stage. This two-stage nested game reflects the sequential dependencies of decisions in each stage. Based on the feedback interaction process, players can capture how to adapt their decisions to maximize their payoffs in an entirely distributed fashion. It is a practical and suitable approach in real-world MCIoT system operations. The main steps of our proposed nested game algorithm are given next and described as a flowchart in Fig. 1.

Step 1: At the initial time, applications in each mobile device are equally partitioned for local and remote offloading computations. At the beginning of the game, this starting guess is useful to monitor the current MCIoT situation.
Step 2: According to Eqs. (1)–(9), mobile devices can estimate their total offload overhead $\left(T\_O_{\text{off}}^{(\cdot)}\right)$ and the total execution overhead $(\mathcal{S}^{(\cdot)}(\mathbb{D}))$ individually.
Step 3: Based on Eqs. (10)–(12), each mobile device adaptively re-partitions its application based on the *Rubinstein-Stahl* bargaining game model.
Step 4: One part is computed locally on the mobile device. In the MC side, the other part is computed remotely on the computation resource, which is selected according to the auction game.
Step 5: For the next resource selection process, computation resources periodically adjust their selling prices $(\psi^{(\cdot)})$ according to Eq. (13).
Step 6: As game players, elastic applications, mobile devices, and MC computation resources are

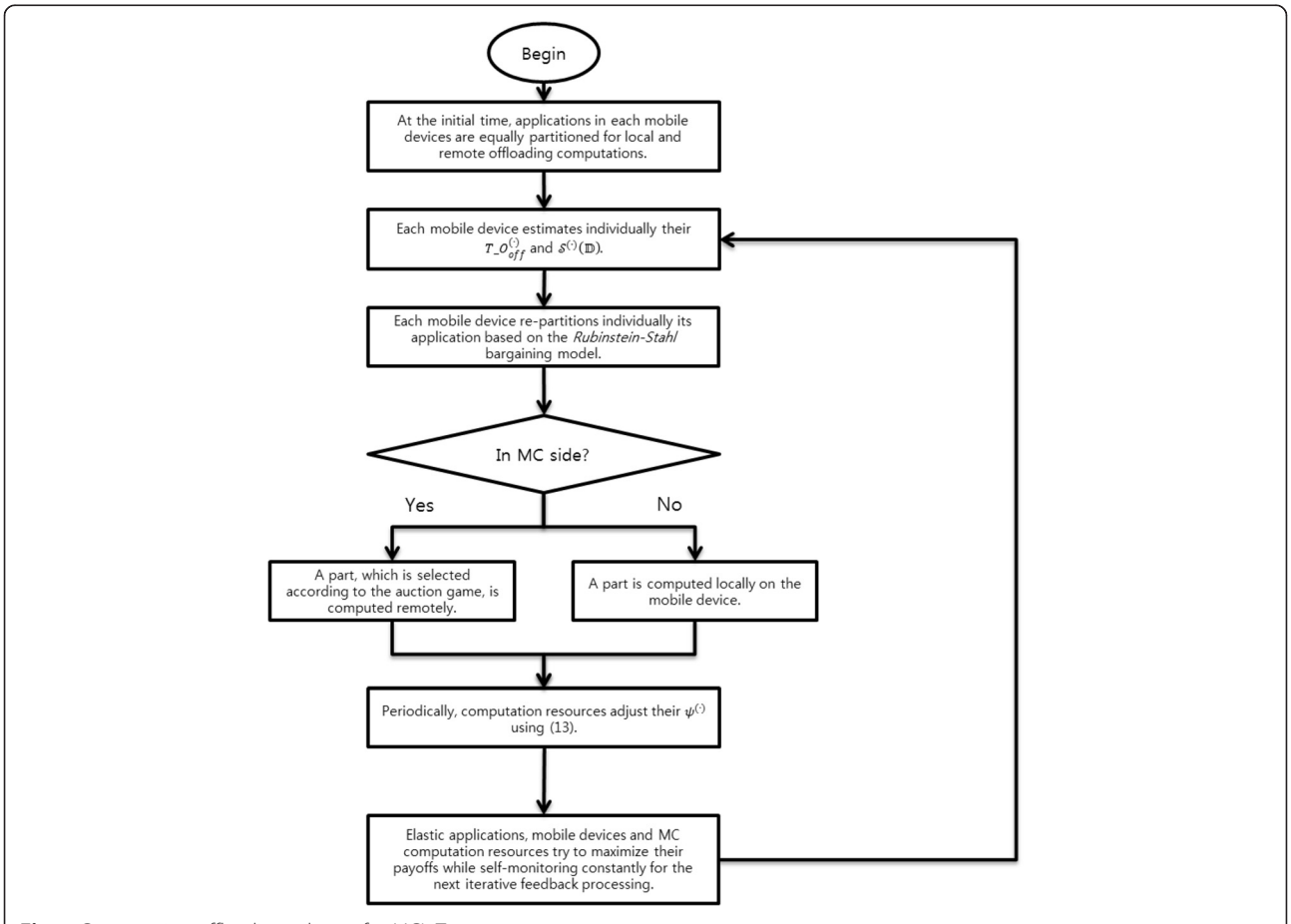

Fig. 1 Computation offloading scheme for MCIoT systems

interrelated and interact with each other in a two-stage nested game. In each stage game, game players try to maximize their payoffs while they are involved in a bigger game.
Step 7: Under widely diverse MCIoT environments, mobile devices and computation resources are self-monitoring constantly for the next iterative feedback processing. This iterative feedback procedure continues under the MCIoT system dynamics.
Step 8: When a new application service is requested, it can re-trigger another computation offloading process; the service proceeds to Step 1 for the next game iteration.

4 Performance evaluation

In this section, the effectiveness of our proposed scheme is validated through simulation. Using a simulation model, the performance of our proposed scheme is compared with two existing computation offload schemes [7, 11]. The assumptions implemented in our simulation model are as follows.

- There are 10 mobile devices and 15 computation resources in our MCIoT system.
- Applications can be split differently according to application type.
- The application generation rate is a Poisson with rate Δ (applications/s), and the range of offered computation load was varied from 0 to 3.0.
- Different applications are assumed based on computation requirement, duration, and required QoS (i.e., T_D).
- The T_D of each application is exponentially distributed with different means for different applications.
- The performance measures obtained on the basis of 50 simulation runs are plotted as a function of the offered applications per second at each mobile device (applications/s/D).
- The MCIoT system performance is estimated in terms of the normalized energy consumption, application execution time, and QoS satisfaction probability under various offered computation loads.

In order to emulate a real MCIoT system environment and for a fair comparison, application types, characteristics, and system parameters are carefully selected for a realistic simulation scenario. Table 1 shows the application types and system parameters used in our simulation.

As mentioned earlier, the *DCOG* and *NTGO* schemes [7, 11] have been recently published and introduced unique challenges to efficiently solve the computation offloading decision problems. However, they are successful only in certain circumstances. In addition, it is observed that the *DCOG* and *NTGO* schemes [7, 11] require higher control overhead for computation offloading via wireless communication. Compared to these schemes, we can confirm the superiority of our proposed approach.

Figure 2 shows the normalized energy consumption of each scheme. To effectively operate the MCIoT system, energy consumption is an important performance metric. All the schemes have similar trends. However, under various offered loads, effective strategic decisions based on

Table 1 Application and system parameters used in the simulation experiment

Application type	Applications	Computation requirement	Computation duration average/sec
I	Voice telephony, video phone	128 K cycle/s	180 s (3 min)
II	Remote login, tele-conference	512 K cycle/s	120 s (2 min)
Parameter	Value	Description	
n	10	The number of mobile devices in MCIoT	
m	15	The number of computation resources in MCIoT	
L	6 or 10	The number of split coded modules in applications	
$\mathscr{F}^{\mathscr{L}}$	64 K cycle/s	A computation capacity of mobile device	
$\mathscr{F}^{\mathscr{R}}$	{128, 256, 512 K cycle/s}	A computation capacity of computation resource	
ρ	1	The price per CPU cycle of mobile device	
$\mathscr{B}$	1 MHz	The channel bandwidth	
P	100 mW	The transmission power of mobile device	
ω	100 dBm	The interference power	
μ	1	The mean of the normal distribution	
σ	1	The standard deviation of the normal distribution	
Parameter	Initial	Description	Values
ψ	1	The price per CPU cycle of resources	Dynamically adjustable
λ	0.5	The relative weights given to time and energy	0 ~ 1

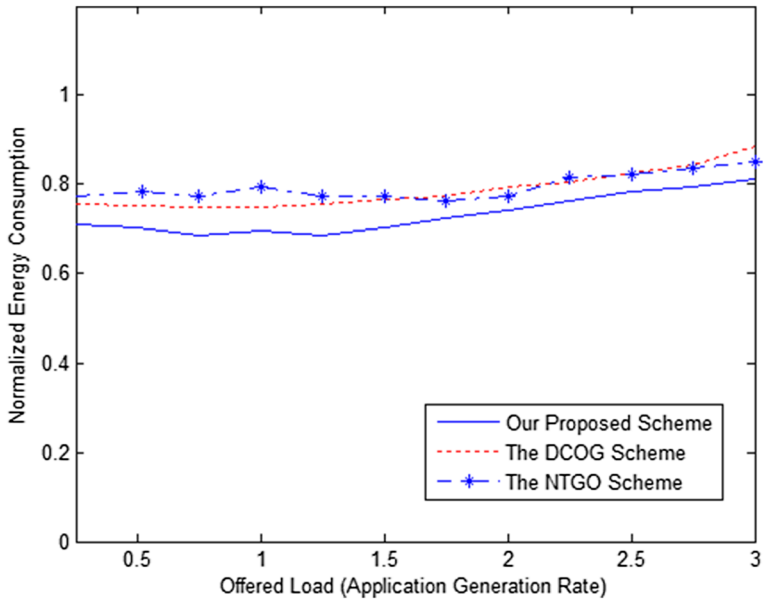

Fig. 2 Normalized energy consumption

our nested game model could lead to higher energy efficiency than the *DCOG* and *NTGO* schemes. When designing an effective computation offloading scheme, it is a highly desirable property.

Figure 3 presents the performance comparison in terms of QoS satisfaction probability. In this work, it is estimated as an application's complete ratio within each deadline. As the offered load in MCIoT system increases, the average amount of available computation resources decreases. Thus, the required QoS of applications is likely to be not satisfied; QoS satisfaction probability decreases. Under widely diverse MCIoT environments, our proposed scheme can provide a higher satisfaction probability for target QoS than the other schemes.

The curves in Fig. 4 show the normalized application execution time in the dynamic MCIoT platform. Usually, computation offloading trades off communication cost for computation gain. The *DCOG* and *NTGO* schemes

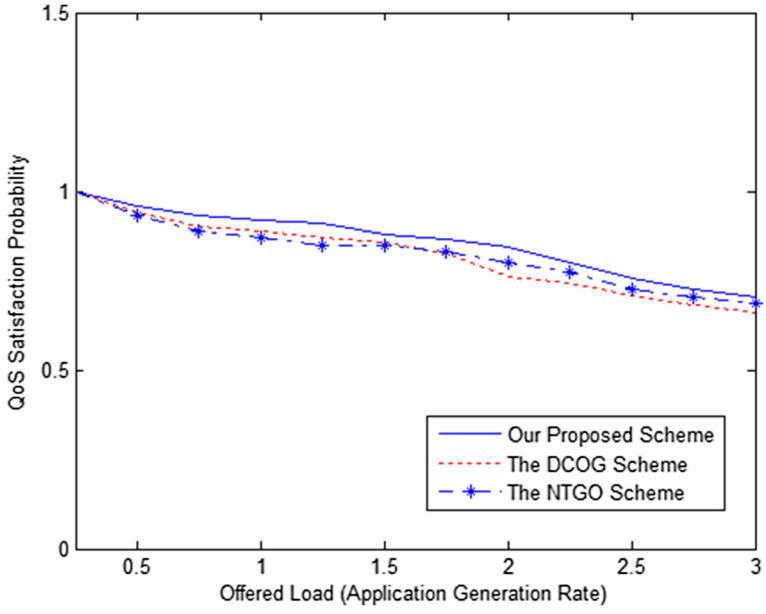

Fig. 3 QoS satisfaction probability

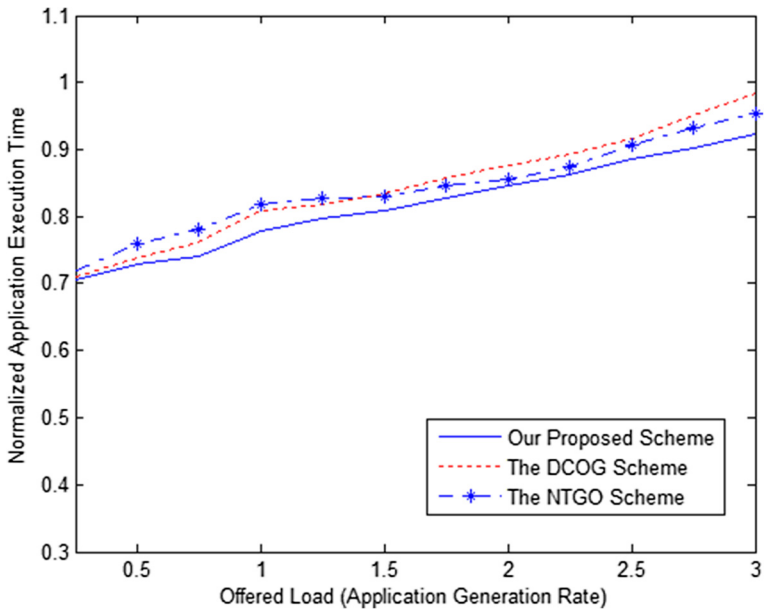

Fig. 4 Normalized application execution time

assume stable network connectivity and adequate cloud computation resources. However, in dynamic system environments, a mobile device may experience communication congestions, while cloud resources may be temporarily unavailable or occupied. Therefore, the communication cost may be higher, while the computation gain will be lower. Moreover, the network and execution time prediction may be inaccurate, causing the performance of MCIoT systems to be degraded. To effectively adapt the unpredictable environment, our scheme constantly monitors the current system conditions. Based on the feedback interaction process, we can maintain a lower application execution time than other existing schemes.

In Fig. 5, the packet loss probabilities are presented. When the offered load is low (below 0.3), the performance of all the schemes is identical. However, as the offered load increases, data packets are likely to be dropped; the packet

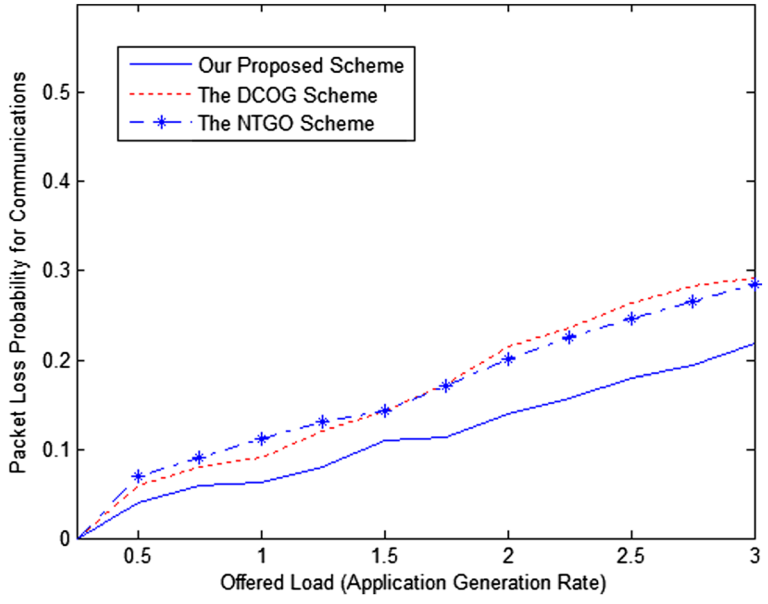

Fig. 5 Packet loss probability for communications

loss probability increases linearly with the offered system load. Under various load intensities, the proposed scheme achieves a lower packet loss rate than other schemes.

From the simulation results in Figs. 2, 3, 4, 5, it can be seen that the proposed scheme, as expected, achieves better performance than the *DCOG* and *NTGO* schemes. Traditionally, energy consumption and total computation time are the key performance indicators for offloading computation. However, there is a trade-off. Based on the nested game model, our approach allows that each mobile device can make decisions individually, while pursuing the minimization of the computation time with a constraint over the energy consumption. It is essential in order to be close to the optimized system performance. Performance evaluation results indicate that our proposed scheme can attain appropriate performance balance, while other schemes [7, 11] cannot offer such an attractive system performance.

5 Summary and conclusions

Over the recent past years, a novel paradigm where cloud and IoT are merged is expected to be an important component of the future Internet. In this paper, we review the integration of MC and IoT and design a new computation offloading scheme in the MCIoT platform. Based on the nested game model, the main goal of our proposed scheme is to maximize mobile device performance while providing service QoS. To satisfy this goal, the proposed nested game model consists of an application partitioning game and cloud resource-selection game. In our partitioning game, applications are adaptively partitioned according to the *Rubinstein-Stahl* bargaining model. In our resource-selection game, computation resources in the MC system are selected based on the one-to-many auction game model. Based on the nested game principle, these game models are interrelated to each other and operated as a two-stage sequential game. The simulation results show that our nested game approach can outperform existing computation offloading schemes. For the future work direction in this promising field, novel system architectures that seamlessly integrate MC and IoT and protocols that facilitate big data streaming from IoT to MC should be addressed when adopting a multi-MCIoT environment. In addition, QoS and QoE, as well as data security, privacy, and reliability issues, are critical concerns for the further research.

Competing interests

The author declares that he has no competing interests.

Acknowledgements

This research was supported by the Ministry of Science, ICT and Future Planning (MSIP), Korea, under the Information Technology Research Center (ITRC) support program (IITP-2015-H8501-15-1018) supervised by the Institute for Information & communications Technology Promotion (IITP) and by the Sogang University Research Grant of 2014(201410020.01).

Received: 20 May 2015 Accepted: 1 October 2015
Published online: 19 October 2015

References

1. KS Kim, S Uno, MW Kim, Adaptive QoS mechanism for wireless mobile network. JCSE **4**(2), 153–172 (2010)
2. A Botta, W de Donato, V Persico, A Pescape, On the integration of cloud computing and internet of things. IEEE FiCloud **2014**, 23–30 (2014)
3. D Singh, G Tripathi, AJ Jara, A survey of internet-of-things: future vision, architecture, challenges and services. IEEE World Forum on Internet of Things (WF-IoT) **2014**, 287–292 (2014)
4. O Vermesan, P Friess, *Internet of Things - Global Technological and Societal Trends from Smart Environments and Spaces to Green Ict*, ed. by O Vermesan, P Friess (River Publishers, Denmark, 2011)
5. AS Sabyasachi, S De, S De, On the notion of decoupling in Mobile Cloud Computing. IEEE HPCC_EUC **2013**, 450–457 (2013)
6. W Zhu, C Lee, A new approach to web data mining based on cloud computing. JCSE **8**(4), 181–186 (2014)
7. C Xu, Decentralized computation offloading game for Mobile Cloud Computing. IEEE Transactions on Parallel and Distributed Systems **26**(4), 974–983 (2015)
8. K Sinha, M Kulkarni, Techniques for fine-grained, multi-site computation offloading. IEEE CCGRID **2011**, 184–194 (2011)
9. Sungwook Kim, Game theory applications in network design, (IGI Global, Hershey, Pennsylvania (USA), 2014)
10. G Tsebelis, Nested games: the cohesion of french electoral coalitions. British Journal of Political Science **18**(2), 145–170 (1988)
11. W Yanzhi, L Xue, M Pedram, A nested two stage game-based optimization framework in Mobile Cloud Computing system. IEEE SOSE **2013**, 494–502 (2013)
12. P Yongshia, S Xie, J Shasha, The application of nested-game theory in the public participation mechanism in the decision-making of large engineering projects. Systems Engineering Procedia **1**, 142–146 (2011)
13. NG Jesse, U Heo, K DeRouen Jr, A nested game approach to political and economic liberalization in democratizing states: the case of South Korea. International Studies Quarterly **46**(3), 401–422 (2002)
14. SS Yau, AB Buduru, Intelligent planning for developing mobile IoT applications using cloud systems. IEEE MS' **2014**, 55–62 (2014)
15. R Hasan, MM Hossain, R Khan, Aura: an IoT based cloud infrastructure for localized mobile computation outsourcing. IEEE MobileCloud **2015**, 183–188 (2015)
16. S Nastic, M Vogler, C Inzinger, T Hong-Linh, S Dustdar, rtGovOps: a runtime framework for governance in large-scale software-defined IoT cloud systems. IEEE MobileCloud **2015**, 24–33 (2015)
17. H Park, M van der Schaar, Bargaining strategies for networked multimedia resource management. IEEE Trans. on Signal Processing **55**(7), 3496–3511 (2007)
18. JE Suris, LA DaSilva, Z Han, AB MacKenzie, Cooperative game theory for distributed spectrum sharing. IEEE ICC **2007**, 5282–5287 (2007)
19. Z Yu, H Zhao, Study on negotiation strategy. International Conference On Power System Technology **2002**, 1335–1338 (2002)
20. M Pan, Y Fang, Bargaining based pairwise cooperative spectrum sensing for cognitive radio networks. IEEE MILCOM **2008**, 1–7 (2008)