Socially Trusted Collaborative Edge Computing in Ultra Dense Networks

Lixing Chen
Electrical and Computer Engineering
University of Miami
Miami, Florida 33146
lx.chen@miami.edu

Jie Xu
Electrical and Computer Engineering
University of Miami
Miami, Florida 33146
jiexu@miami.edu

ABSTRACT

Small cell base stations (SBSs) endowed with cloud-like computing capabilities are considered as a key enabler of edge computing (EC), which provides ultra-low latency and location-awareness for a variety of emerging mobile applications and the Internet of Things. However, due to the limited computation resources of an individual SBS, providing computation services of high quality to its users faces significant challenges when it is overloaded with an excessive amount of computation workload. In this paper, we propose collaborative edge computing among SBSs by forming SBS coalitions to share computation resources with each other, thereby accommodating more computation workload in the edge system and reducing reliance on the remote cloud. A novel SBS coalition formation algorithm is developed based on the coalitional game theory to cope with various new challenges in small-cell-based edge systems, including the co-provisioning of radio access and computing services, cooperation incentives, and potential security risks. To address these challenges, the proposed method (1) allows collaboration at both the user-SBS association stage and the SBS peer offloading stage by exploiting the ultra dense deployment of SBSs, (2) develops a payment-based incentive mechanism that implements proportionally fair utility division to form stable SBS coalitions, and (3) builds a social trust network for managing security risks among SBSs due to collaboration. Systematic simulations in practical scenarios are carried out to evaluate the efficacy and performance of the proposed method, which shows that tremendous edge computing performance improvement can be achieved.

CCS CONCEPTS

• **Networks** → **Mobile networks**; • **Computer systems organization** → *Cellular architectures*; *Client-server architectures*; *Client-server architectures*;

KEYWORDS

Edge computing, coalitional game, ultra dense small-cell network

SEC '17, San Jose / Silicon Valley, CA, USA

DOI: 10.1145/3132211.3134451

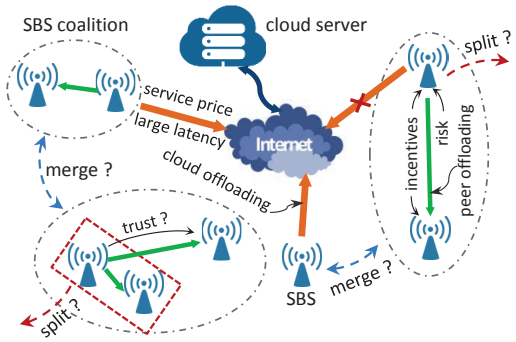


Figure 1: Illustration of socially trusted collaborative EC

1 INTRODUCTION

Pervasive mobile computing and the Internet of Things are driving the development of many new applications that are both compute-demanding and latency-sensitive, such as cognitive assistance, mobile gaming and augmented reality. Although cloud computing enables convenient access to a centralized pool of configurable and powerful computing resources, it often cannot meet the stringent requirements of latency-sensitive applications due to the often unpredictable network latency and expensive bandwidth [14, 25]. The growing amount of distributed data further makes it impractical or resource-prohibitive to transport all the data over today's already-congested backbone networks to the remote cloud [20]. As a remedy to these limitations, Edge Computing (EC) [8] emerges as a new computing paradigm to push the frontier of computing applications, data, and services away from centralized cloud computing infrastructures to the logical extremes of a network thereby enabling analytics and knowledge generation to occur at the data source.

Considered as a key enabler of EC, small cell base stations (SBSs), such as femtocells and picocells, endowed with cloud-like computing and storage capability can serve end-users' computation requests as a substitute of the cloud while providing LTE connectivity to the Internet, especially for indoor premises. Such migration between computation and wireless communication service provisioning at the network edge was envisioned in the TROPIC project [27]. Nonetheless, compared with mega-scale data centers, SBS will be limited in computing resources. Therefore, computation workload exceeding the SBS's computing capacity still has to be sent to the cloud, resulting in a hierarchical offloading structure among end-users, SBSs, and the cloud. Although there have been a few works [28] studying the optimization of the offloading strategy in this hierarchical structure, relying on a single SBS significantly

limits the EC performance. Fortunately, the ultra dense deployment of SBSs in the next generation (5G) mobile networks [1] creates an opportunity for nearby SBSs to collaboratively form a computation resource pool. Such a collaborative EC infrastructure, also known as Femto-Cloud [26], improves the efficiency of system resource utilization by exploiting the spatial diversity of workload patterns, thereby significantly enhancing EC performance. For instance, a cluster of SBSs can coordinate among themselves to serve computation requests by transferring workload from overloaded SBSs to nearby SBSs with a light workload.

The idea of balancing computation workload among nearby SBSs is similar to geographical load balancing [12, 13] in data center networks, which has been extensively studied. However, edge computing in ultra dense networks faces many new challenges of forming an effective collaborative network. First, whereas conventional clouds manage only the computing resource, moving the computing resource to the network edge leads to the co-provisioning of radio access and computing services by the SBSs, thus mandating a new model for understanding the interdependency between the management of the two resources. Unlike cloud computing that is agnostic to user location, the physical association between the SBSs and the end users becomes a critical design aspect that has a significant impact on the computing performance. Second, a distinct feature of SBSs is that they are often owned and deployed by individual users (e.g. home/enterprise owners). Unlike macro base stations that are deployed by the network operator, the operator has only minimum control over SBSs. Without proper incentives, SBSs will be reluctant to participate in the collaborative edge computing process. Therefore, incentives must be devised and incorporated into the load balancing scheme. Of particular importance is the design of a trust management module that takes into account the social trust relationship between the SBSs to minimize the security and privacy risks in collaboration.

In this paper, we design a socially trusted collaborative edge computing platform for ultra dense networks (see Figure 1 for illustration). (1) We use payment as the incentive mechanism for individual SBSs to collaborate. Specifically, overloaded SBSs can pay nearby SBSs with spare computing resources to process their workload instead of offloading it to the remote cloud. The collaborative network formation and the associated payment scheme are designed under the coalitional game theoretic framework and we prove that our proposed scheme results in the optimal stable coalition among the SBSs. (2) When forming the coalition among SBSs, in addition to workload balancing at the SBS level, we allow end-users to directly switch their association to neighboring SBSs by exploiting the ultra dense deployment of SBSs, thereby avoiding transmission energy and latency incurred in the workload transfer between SBSs. This mixed workload balancing scheme is in stark contrast with geographical load balancing in data center networks or collaborative Femto-Cloud [26] that does not consider the ultra dense deployment. (3) The proposed platform has a dedicated trust management component that manages the trust between any two SBSs to enable security-aware collaboration. Apart from the physical network of SBSs, we construct a social trust network that characterizes the trust between SBSs, which can be used to determine the security measure to be put on the processing/offloading of computation workload from/to neighbor SBSs, thereby reducing security and privacy leakage risks. (4) We conduct extensive systematic simulation studies to evaluate our proposed scheme in practical settings and understand how different SBS coalitions are formed and when collaborative EC is the most useful. Our results show that the proposed collaborative EC method can reduce the system cost by more than 40% compared to standalone EC systems without collaboration.

The rest of this paper is organized as follows. Section 2 presents the system model. Section 3 develops a coalitional game for cooperative SBS network. Section 4 proposes a merge-split framework for distributed coalition formation. Simulations are carried out in Section 5. Section 6 reviews related works, followed by the conclusion in Section 7.

2 SYSTEM MODEL

We consider N SBSs, indexed by $\mathcal{N} = \{1, 2, ..., N\}$, endowed with heterogeneous computing capabilities. SBSs are located in separate rooms, possibly on different floors, in a multi-story building. These SBSs have Internet access and therefore can offload computation tasks to the remote cloud when its own computational capacity cannot accommodate the demand. Let $\mathcal{M} = \{1, 2, ..., M\}$ denote the set of all mobile user equipments (MUEs) in the building. Each SBS has a set of authorized MUEs, denoted by $\mathcal{M}_i \subseteq \mathcal{M}$. For instance, MUEs (e.g. mobile phones, laptops etc.) of employees in a business are authorized to access the communication/computing service of the SBS deployed by the business. MUEs of family members can also access its home SBS. However, due to the ultra dense deployment of SBSs in the building, an MUE is in the radio coverage of multiple SBSs and hence, there is a potential for SBSs to collaboratively balance the workload via direct MUE-SBS association.

We consider a time-slotted system. In each time slot, each MUE has a certain amount of computationally intensive tasks that need be offloaded to either the SBSs or the cloud for processing. We will focus on the problem in one time slot. There are two types of computation tasks: private tasks and normal tasks. Private tasks of an MUE must be processed by its own SBS or the cloud (which is assumed to be secure). They cannot be processed by other SBSs due to privacy concerns since SBSs are deployed by individual home/business owners which may not be fully trusted and less protected than the cloud. Normal tasks are less security-sensitive and hence, they can be processed by either its own SBS, the secure cloud or other nearby SBSs. Therefore, the task requests from MUE $m \in \mathcal{M}$ in a time slot are described by a tuple (λ_m^a, τ_m) where λ_m^a is rate of task arrival in the current time slot (assuming a Poisson arrival process) and $\tau_m \in [0, 1]$ is the fraction of private tasks. Without SBS collaboration, the total task arrival rate to SBS i from all its authorized MUEs is thus:

$$\lambda_i^s = \sum_{m \in \mathcal{M}_i} \lambda_m^a \tag{1}$$

Due to the limited computational capacity of SBS i, there is a maximum task arrival rate $\omega_i^{\max}$ that SBS i can handle. We define $\alpha_i \triangleq \omega_i^{\max} - \lambda_i^s$ as the computing resource surplus (or deficit) of SBS i. If $\alpha_i \geq 0$, then SBS i has spare computing resources that it can share with other SBSs. If $\alpha_i < 0$, then SBS i needs to acquire

additional computing resource from either the cloud or other peer SBSs to meet its demand. Therefore, there is a potential computing resource exchange market among the SBSs. We call SBSs with $\alpha \geq 0$ potential "sellers" and those with $\alpha < 0$ potential "buyers".

2.1 Overview of Collaborative Edge Computing

Buyer SBSs have the following two options to acquire additional computing resources:

SBS-to-Cloud offloading: The SBS can further offload the unsatisfied computation tasks to the remote cloud. However, there will be extra transmission delay costs due to the large round-trip time to the remote cloud. Moreover, the cloud will also charge the SBS service fees for using the cloud computing resources.

SBS coalitions: Instead of relying on the remote cloud, SBSs can collaborate with each other by forming edge computing coalitions, in order to reduce the number of computation tasks offloaded to the cloud, thereby improving the delay performance and cutting their expenses on using the cloud service. Specifically, there are two ways to collaborate:

- *SBS peer offloading*: a buyer SBS first receives the computation tasks from its authorized MUEs and then further transmits via the wireless link some of the received tasks to nearby seller SBSs for processing.
- *MUE-SBS association*: if an authorized MUE is also in the coverage of the seller SBS, a buyer SBS can associate this MUE to the seller SBS and directly offload its computation tasks to that seller SBS. This method exploits the dense deployment of SBSs and further saves the task transmission delay and energy cost.

Clearly, load balancing via MUE-SBS association is preferred due to the reduced overhead cost. Therefore, in SBS coalitions, MUE-SBS association is adopted with a higher priority. When load balancing cannot be realized via MUE-SBS association, SBS peer offloading is then executed. Typically, these two methods are used at the same time since MUEs are distributed in the network and not all MUEs are covered by seller SBSs.

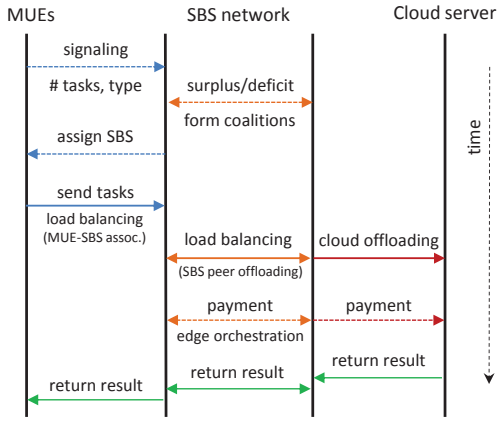


Figure 2: Operating time sequence diagram of collaborative edge computing

Different coalitions (i.e. seller-buyer matchings) lead to different edge computing performance. Since SBSs are self-interested and do not provide computing services to other SBSs for free, a key issue that this paper will address is how to design payment mechanisms to provide seller SBSs with incentives to cooperate and form the optimal stable coalitions. Before we design the coalition mechanism, we first show a time sequence diagram in Figure 2 to illustrate the operation of the collaborative edge computing system. At the beginning of each operational slot, a signaling phase is introduced in which MUEs report task requests. Based on these requests, SBSs form edge computing coalitions by playing a distributed coalitional game, taking into consideration of the local processing cost, cloud offloading cost, SBS peer offloading cost and collaboration risk. Buyer SBSs make payments to seller SBSs for using their computing services. An edge orchestrator, which is a trusted third party, is introduced to facilitate the payment process. Specifically, the buyer SBSs first send payments to the edge orchestrator, which then distributes the payments to the seller SBSs. The time slot ends with returning computation results to the MUEs.

Next, we model the various costs involved in the collaborative edge computing system. For each SBS, its total cost comprises two main parts: operational cost and risk management cost. These costs form the basis of the coalition formation game.

2.2 Operational Cost

We first model the operational cost of the edge computing coalitions incurred in different stages of the system.

Stage I: MUE-SBS association. We consider that an SBS absorbs the association cost (i.e. MUE-to-SBS transmission delay and energy consumption) of its authorized MUEs as part of its operational cost. Let $\mathcal{M}_{ij} \subseteq \mathcal{M}_i (\forall j \in \mathcal{N}, j \neq i)$ denote the set of authorized MUEs of SBS i that are associated to SBS j. The achievable transmission rate r^a_{mi} between MUE m and SBS i is given by the Shannon capacity

$$r^a_{mi} = W \log\left(1 + \frac{p^a_{mi} H_{mi}}{\sigma^2 + I}\right) \tag{2}$$

where W is the channel bandwidth, H_{mi} is the channel gain between SBS i and user m, σ^2 is the noise power and I is the interference from other SBSs. Given a target transmission rate r^a_{mi}, the transmission power is thus

$$p^a_{mi} = (2^{\frac{r^a_{mi}}{W}} - 1)(\sigma^2 + I) H^{-1}_{mi} \tag{3}$$

To simplify the notations, we assume that the expected data size of each task is a unit size. Therefore, the expected transmission delay and energy consumption of each task are $d^a_{mi} = 1/r^a_{mi}$ and $e^a_{mi} = p^a_{mi}/r^a_{mi}$, respectively. The MUE-SBS association cost of SBS i is therefore

$$\begin{aligned} C^a_i = & \sum_{m \in M_{ii}} \lambda^a_m (d^a_{mi} + \gamma e^a_{mi}) \\ & + \sum_{j \in \mathcal{N}} \sum_{m \in \mathcal{M}_{ij}} (1 - \tau_m) \lambda^a_m (d^a_{mj} + \gamma e^a_{mj}) \\ & + \sum_{j \in \mathcal{N}} \sum_{m \in \mathcal{M}_{ij}} \tau_m \lambda^a_m (d^a_{mi} + \gamma e^a_{mi}) \end{aligned} \tag{4}$$

where γ is a normalization coefficient for delay cost and energy cost. The first term on the right-hand side is the association cost of authorized MUEs of SBS i that only associate with SBS i. The second and third terms are the costs due to authorized MUEs of

SBS i associating with other SBSs $j \neq i$. However, for each MUE m, its private tasks must be sent to its own SBS i. Note that for a seller SBS i, we must have $\mathcal{M}_{ij} = \emptyset, \forall j$ since it has enough computing resources to accommodate all task requests from its own authorized MUEs.

Stage II: SBS peer offloading. In this stage, the cost of SBSs is mainly caused by task migration between SBSs due to transmission delay and energy consumption. Let β_{ij} denote the amount of tasks sent from SBS i to SBS j. The transmission rate from SBS i to SBS j and the associated transmission power are denoted by r_{ij} and p_{ij}, respectively, which can be derived in a similar way as in Stage I. Therefore, the transmission delay and energy consumption for each task is $d_{ij}^{s,tx} = 1/r_{ij}$ and $e_{ij}^{s,tx} = p_{ij}/r_{ij}$. The transmission cost incurred to SBS i due to peer offloading is therefore:

$$C_i^{s,tx} = \sum_{j \in \mathcal{N}, j \neq i} \beta_{ij} \left(d_{ij}^{tx} + \gamma e_{ij}^{tx} \right) \tag{5}$$

Stage III: SBS computing. We then model the cost for processing computation tasks locally at SBSs. For each task, we assume that the required number of CPU cycles is an exponential random variable with mean ρ. The computational capability of SBS i is measured by its CPU speed (i.e. CPU cycles per second), denoted by f_i. We model the computing delay using the M/M/1 queuing system. Thus the average computation delay (including task waiting time and processing time) for each task, can be obtained as

$$d_i^{s,c}(\omega_i) = \frac{1}{f_i/\rho - \omega_i} \tag{6}$$

where ω_i is the workload processed at SBS i, which is the outcome of SBS coalition and load balancing and will be discussed shortly. The computation energy consumption for each task processed at SBS i is proportional to the square of the CPU speed $(f_i)^2$, presented as $e_i^{s,c} = \kappa (f_i)^2$, where κ is a constant depending on the CPU architecture [14]. Therefore, the computation cost of SBS i is

$$C_i^{s,c} = \omega_i (d_i^{s,c}(\omega_i) + \gamma e_i^{s,c}) \tag{7}$$

Notice that although here we use specific functions for computing delay and energy consumption, other functions can also be adopted. In practice, an SBS may maintain a lookup table for the expected delay and energy consumption under different workload inputs.

Stage IV: SBS-to-Cloud offloading. SBSs may still have to offload some computation tasks to the cloud. Let β_{i0} be the number of computation tasks offloaded to the cloud by SBS i. Due to the large round-trip time to the remote cloud and the transmission energy consumption, the SBS-to-Cloud offloading cost of SBS i is

$$C_i^{c,tx}(\beta_{i0}) = \beta_{i0}(d_{i0}^{c,tx} + \gamma e_{i0}^{c,tx}) \tag{8}$$

where $d_{i0}^{c,tx}$ is the total expected delay (due to both transmission and computation on the cloud) from SBS i to the cloud, and $e_{i0}^{c,tx}$ is the transmission energy consumed by SBS i for each task. In addition, the cloud charges SBSs w_0 \$/task for using the cloud service. Therefore, SBS i will also incur an monetary cost

$$M_i = w_0 \beta_{i0} \tag{9}$$

To sum up, the total operational cost of SBS i is:

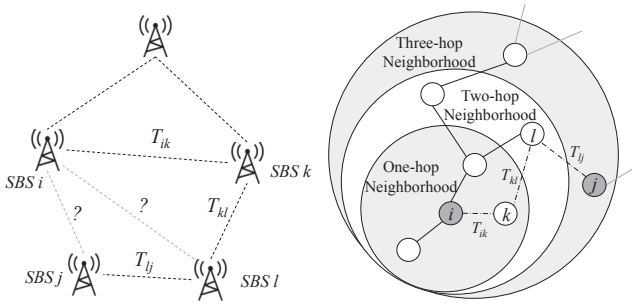
$$C_i = w_c (C_i^a + C_i^{s,tx} + C_i^{s,c} + C_i^{c,tx}) + M_i \tag{10}$$

where w_c converts the delay and energy cost into a value comparable to the monetary cost. Keep in mind that this cost depends on how the SBSs form the coalition and balance the workload among themselves.

2.3 SBS Social Trust Model

Since SBSs are operated by individual owners, there are higher security and privacy risks for SBSs to offload their tasks to other SBSs than processing locally or offloading to the secure cloud. Therefore, when forming SBS coalitions, trust between SBSs must be taken into account.

Apart from the physical network of SBSs, we define a social trust network that describes the trust relationships between SBSs. Let $T_{ij} \in [0, 1]$ denote the trust value that SBS i assigns to SBS j. $T_{ij} = 0$ indicates that SBS i completely distrusts SBS j and $T_{ij} = 1$ indicates that SBS i completely trusts SBS j. However, even if two SBSs are neighbors in the physical network, they may not have an established trust relationship between each other. For instance, a new SBS may have just been set up or two SBSs have not interacted with each other for a long time. As illustrated in Figure 3, although SBS i and SBS j are physical neighbors and hence can potentially form a coalition, the values of T_{ij} and T_{ji} are unknown since they do not have a social relationship.



(a) Physical network (b) Social trust network

Figure 3: Physical and social network.

In such cases, trust between the SBSs will be derived using the social trust network. If SBS i can reach SBS j via certain other SBSs in the social network, then the trust between SBS i and SBS j is computed by propagating trust along the path that connects them. Let $\Lambda(i, j)$ be the shortest path between SBS i and SBS j, then trust T_{ij} is computed as

$$T_{ij} = \prod_{(k,l) \in \Lambda(i,j)} T_{kl} \tag{11}$$

We note that the above is just one common way to evaluate trust using the social trust network. There are many other ways to evaluate trust proposed in the literature. Moreover, the trust values will be updated over time depending on the recent social interactions (including interactions in the coalitions) among SBSs.

Using the social trust network, the cost due to managing the security risk that SBS i faces when offloading computation tasks to

other SBSs (e.g. adopting stronger yet more costly security mechanisms) is modeled as follows:

$$R_i = w_r \sum_{j \in \mathcal{N}} (\beta^u_{ij} + \beta_{ij})(1 - T_{ij}) \tag{12}$$

where $\beta^u_{ij} = \sum_{m \in \mathcal{M}_{ij}} (1 - \tau_m)\lambda^a_m$ is the amount of offloaded workload due to direct MUE-SBS association, β_{ij} is the amount of offloaded workload in SBS peer offloading and w_r converts risk into a value comparable to the monetary cost.

3 COLLABORATIVE SBS NETWORK AS A COALITIONAL GAME

To formally study the formation of SBS coalitions and the resulting workload offloading decisions, we use the framework of the coalitional game theory [2, 5]. A coalitional game is defined as a tuple $(\mathcal{N}, v)$ where $\mathcal{N}$ is the player's set and $v : 2^{\mathcal{N}} \rightarrow \mathbb{R}$ is a function that assigns for every possible coalition $S \subseteq \mathcal{N}$ a real number representing the total benefit achieved by coalition S. By evaluating the values of different coalitions, players decide what coalitions are formed among themselves. In what follows, we first define the value function $v(S)$ for any given coalition $S \subseteq \mathcal{N}$. Clearly, $v(S)$ depends on not only which SBSs are in the coalition S but also how they collaborate, namely how they perform load balancing. Then we will describe what coalitions are desired in terms of stability.

3.1 Collaborative load balancing and value function for coalitions

In this subsection, we investigate the interaction among SBSs that belong to any given coalition S (which may not be stable though). Let $S_s \subset S$ denote the set of seller SBSs in S and $S_b \subset S$ denote the set of buyer SBSs in S. We must have $S_s \cup S_b = S$. Although there are many approaches to match buyers with sellers in the coalition (e.g., double auction or other matching algorithms [22]), for ease of implementation, we consider a simple scheme similar to [23] that relies on the preference of the buyers inside the coalition to decide the workload offloading decisions among the SBSs. Specifically, buyers can act sequentially (according to some order Π) to acquire their needed computation resource as follows:

(1) Buyer $b_i \in S_b$ requests to acquire its needed computation resource from seller $s_j \in S_s$ that will potentially yield the smallest offloading cost according to the SBS peer offloading scheme in Algorithm 1 (which returns results of $\mathcal{M}_{b_i s_j}$ and $\beta_{b_i s_j}$).
   - If seller s_j can offer the required computation resource of buyer b_i, then the buyer does not act further.
   - Otherwise, buyer b_i acquires as much computation resource as possible from seller s_j and then tries to fulfill the rest of its computation resource demand by acquiring resources from other sellers in S_s.

(2) Buyer b_i repeats the above sequence until it has covered up all its computation resource deficit or no available sellers in S_s exist. Then the next buyer SBS starts its acquiring process.

This process is repeated for all buyers in S_b. If a buyer is unable to find a seller in S_s and still needs computation resource, then the buyer will offload the remaining computation tasks, represented by $\beta_{b_i 0}$ to the cloud. Essentially a buyer SBS tries to offload as much computation workload as possible to peer seller SBSs.

Algorithm 1 SBS peer offloading scheme

Input: Computation surplus $|\alpha_{s_j}|$ of seller s_j; Computation deficit $|\alpha_{b_i}|$ of buyer b_i.

Maximum task offloading: $\alpha = \min(|\alpha_{b_i}|, |\alpha_{s_j}|)$;
Stage 1: *MUE-SBS association*
Find users co-covered by b_i and s_j, denoted by $\mathcal{R}_{b_i s_j}$;
if $\sum_{m \in \mathcal{R}_{b_i s_j}} (1 - \tau_m)\lambda^a_m \geq \alpha$ **then**
    Choose $\mathcal{M}_{b_i s_j}$: $\sum_{m \in \mathcal{M}_{b_i s_j}} (1 - \tau_m)\lambda^a_m = \alpha$; **Stop**;
else
    $\mathcal{M}_{b_i s_j} = \mathcal{R}_{b_i s_j}$;
    $\tilde{\alpha} = \alpha - \sum_{m \in \mathcal{M}_{b_i s_j}} (1 - \tau_m)\lambda^a_m$; **Go to Stage 2**;
end if
Stage 2: *SBS peer offloading*
$\beta_{b_i s_j} = \tilde{\alpha}$;
return $\mathcal{M}_{b_i s_j}, \beta_{b_i s_j}$;

Following the above buyer-seller matching and workload allocation process, the values of $\mathcal{M}_{b_i s_j}$, $\beta_{b_i s_j}$ and $\beta_{b_i 0}$ can thus be determined for each b_i and s_j. In particular, the workload ω_{b_i} (or ω_{s_j}) that buyer SBS b_i (or seller SBS s_j) needs to process locally can be determined as follows:

$$\omega_{b_i} = \lambda^s_{b_i} - \sum_{s_j \in S_s} \sum_{m \in \mathcal{M}_{b_i s_j}} (1 - \tau_m)\lambda^a_m - \sum_{s_j \in \{S_s \cup \{0\}\}} \beta_{b_i s_j} \tag{13}$$

$$\omega_{s_j} = \lambda^s_{s_j} + \sum_{b_i \in S_b} \sum_{m \in \mathcal{M}_{b_i s_j}} (1 - \tau_m)\lambda^a_m + \sum_{b_i \in S_b} \beta_{b_i s_j} \tag{14}$$

With all these values derived, we are able to compute the operational cost C_i as well as the risk management cost R_i for each SBS $i \in S$ according to our system model. Clearly, the values of these costs also depend on the ordering of the buyer SBSs in the aforementioned matching process. Let O_S be the set of all possible orderings over buyers in S. Then given an ordering $\Pi \in O_S$, the utility of SBS i in coalition S is thus defined as

$$u_i(S, \Pi) = -(C_i + R_i), i \in S \tag{15}$$

and the total utility for coalition S is

$$U(S, \Pi) = \sum_{i \in S} u_i(S, \Pi) \tag{16}$$

The minus sign is inserted to turn the problem into a maximization problem in order to facilitate the analysis of the coalitional game. The value function for the SBSs coalition formation game $(\mathcal{N}, v)$ is defined as

$$v(S) = \max_{\Pi \in O_S} U(S, \Pi) \tag{17}$$

which is the maximum achievable total utility over all possible orderings of the buyers, or equivalently, the minimum achievable total cost of SBSs in the coalition.

3.2 Payment scheme within a coalition

The previous subsection describes how SBSs in a coalition can collaborate with each other to improve their total utility (i.e. reduce their total cost). However, although the overall performance of the coalition may be improved, individual SBSs, especially the seller

SBSs, do not share their computing resources with others for free. In this subsection, we design a payment scheme to provide seller SBSs with incentives to cooperate. In the next section, we will study what stable coalitions are formed under this payment-based incentive mechanism.

Let g_i be the payment/reward of a buyer/seller SBS i, then its post-payment utility ϕ_i becomes

$$\phi_i = u_i - g_i \tag{18}$$

Clearly, the total payment must equal the total reward within a coalition and hence $\sum_{i \in S} g_i = 0$. If $g_i > 0$, then SBS i pays g_i. If $g_i < 0$, then SBS i receives $|g_i|$ reward.

Our payment scheme is developed by following a simple yet strict fairness criterion, namely *proportional fairness payoff division*. Nevertheless, other fairness criteria, such as egalitarian fair, Shapley value, nucleolus, can also be adopted. In this scheme, the values of payments and rewards are decided by dividing the payoff (the utility improvement) of the whole coalition due to cooperation among the SBSs proportionally to their utility achieved without cooperation. Specifically, for SBS i, its post-payment utility will be

$$\phi_i = \psi_i \left(v(S) - \sum_{j \in S} v(\{j\}) \right) + v(\{i\}) \tag{19}$$

where $v(\{i\})$ represents the utility of SBS i if it does not join any coalition (so only local processing or offloading to the cloud), and ψ_i is the proportional weight satisfying $\sum_{i \in S} \psi_i = 1$ and

$$\frac{\psi_i}{\psi_j} = \frac{\tilde{v}(\{i\})}{\tilde{v}(\{j\})} \tag{20}$$

where $\tilde{v}(\{i\})$ is the normalized utility of SBS i within its coalition. The normalization is to map negative SBS utilities to a positive interval. It is easy to verify that $\sum_{i \in S} \phi_i = v(S)$. Based on the proportional fairness criterion, the payment/reward of SBS i can thus be determined as

$$g_i(S) = \phi_i(S) - u_i(S), \forall i \in S \tag{21}$$

In the above equation, ϕ_i can be interpreted as the expected utility of SBS i by participating in the coalition, while the u_i is the actually realized utility. The gap of the two is filled by the payment scheme.

There are two implementation issues for the payment scheme. First, payments need to be properly distributed since multiple buyers and sellers may be involved in the transaction. Moreover, direct payment from buyers to sellers faces fraud risks in the monetary transaction. To enable the effective and safe transaction, the edge orchestrator, which is a trusted third-party, collects payments from all buyers and then distributes them to the sellers.

3.3 Stability of Coalitions

SBSs may form multiple disjoint coalitions and there are many ways that SBSs form coalitions. However, we are interested in forming *stable* coalitions such that no SBS or group of SBSs have incentives to leave the current coalition to form a different coalition. In particular, the requirement that all SBSs (buyers and sellers) in a coalition must at least receive higher utilities than working individually is a necessary but not sufficient condition for stability.

Consider any subset $\mathcal{K} \subseteq \mathcal{N}$ of SBSs, we call $\mathcal{S} = \{S_1, ..., S_L\}$ a collection of coalitions formed by these SBSs, where $S_l \subseteq \mathcal{K}, \forall l$ are disjoint subsets of $\mathcal{K}$. If $\mathcal{K} = \mathcal{N}$, then we call $\mathcal{S}$ a partition of $\mathcal{N}$. A defection function $\mathbb{D}$ is a function that associates each possible partition $\mathcal{S}$ of $\mathcal{N}$ with a group of collections. The stability of a partition $\mathcal{S}$ is defined with respect to a defection function.

Definition 3.1. ($\mathbb{D}$-**stability**). A partition $\mathcal{S}$ of $\mathcal{N}$ is $\mathbb{D}$-stable if no group of SBSs are interested in leaving $\mathcal{S}$ and forming a new collection of coalitions $\mathcal{S}' \in \mathbb{D}(\mathcal{S})$. That is, at least one SBS in such a group does not improve its utility by leaving the current partition.

In other words, a defection function $\mathbb{D}$ restricts the possible ways that SBSs may deviate/defect. Two defection functions are of particular interest. The first function, denoted by $\mathbb{D}_c$, associates with each partition $\mathcal{S}$ the group of all possible collections in $\mathcal{N}$, namely there is no restriction on the way SBSs may deviate. The second function, denoted by $\mathbb{D}_{hp}$, associates each partition $\mathcal{S}$ with the group of collections that can be formed by merging or splitting coalitions in $\mathcal{S}$. Therefore, $\mathbb{D}_{hp}$-stability is weaker than $\mathbb{D}_c$-stability. In the next section, we design a distributed SBS coalition formation algorithm that achieves at least $\mathbb{D}_{hp}$-stability.

4 DISTRIBUTED COALITION FORMATION

4.1 Distributed Coalition Formation Algorithm

To present the distributed SBS coalition formation algorithm, we first introduce the notion of Pareto dominance to compare the "quality" of two collections of coalitions.

Definition 4.1. (**Pareto Dominance**) Consider two collections of disjoint coalitions $\mathcal{S}_1$ and $\mathcal{S}_2$ formed by the same subset of SBSs $\mathcal{K} \subseteq \mathcal{N}$. $\mathcal{S}_1$ Pareto-dominates $\mathcal{S}_2$, denoted by $\mathcal{S}_1 \rhd \mathcal{S}_2$, if and only if $\phi_i(\mathcal{S}_1) \geq \phi_i(\mathcal{S}_2), \forall i \in \mathcal{K}$ with at least one strict inequality for some SBS.

Pareto dominance implies that a group of SBSs prefer to form coalitions in $\mathcal{S}_1$ rather than $\mathcal{S}_2$, if and only if at least one SBS is able to strictly improve its utility without hurting any other SBS. The following two operations, namely *merge* and *split* [2], are central to our coalition formation algorithm:

- **Merge**: merge a set of coalitions $\{S_1, \ldots, S_l\}$ into a bigger coalition $\bigcup_{j=1}^{l} S_j$ if $\{\bigcup_{j=1}^{l} S_j\} \rhd \{S_1, \ldots, S_l\}$.
- **Split**: split a coalition $\{\bigcup_{j=1}^{l} S_j\}$ into a set of smaller coalitions $\{S_1, \ldots, S_l\}$ if $\{S_1, \ldots, S_l\} \rhd \{\bigcup_{j=1}^{l} S_j\}$.

By performing Merge, a group of SBSs can operate and form a single and larger coalition if this formation increases the utility of at least one SBS without decreasing the utility of any other involved SBSs. Hence, a Merge decision ensures that all involved SBSs agree on its occurrence. Likewise, a coalition can decide to split and divide itself into smaller coalitions if splitting is preferred in the Pareto sense.

Our SBS coalition formation algorithm is developed based on the Merge and Split operations, which is presented in Algorithm 2. The algorithm consists of two phases. The coalition formation phase iteratively executes the Merge and Split operations. Given the current partition $\mathcal{S}$, each coalition $S \in \mathcal{S}$ negotiates, in a pairwise manner, with neighboring SBSs to assess a potential merge.

Algorithm 2 Distributed SBS coalition formation

Initial: The SBS network is partitioned by $\mathcal{S} = \mathcal{N} = \{1, \ldots, N\}$ with non-cooperative SBSs at the beginning of each operational time slot.
Phase 1: SBS Coalition Formation
Repeat
(a) $\mathcal{S}' \leftarrow$ Merge($\mathcal{S}$): SBS coalitions in $\mathcal{S}$ decide to merge by examining the Pareto dominance.
(b) $\mathcal{S} \leftarrow$ Split($\mathcal{S}'$): SBS coalitions in $\mathcal{S}'$ make distributed split decision using the Pareto dominance.
Until Merge and split converges to a final partition $\mathcal{S}_f$
Phase 2: Cooperative computation offloading
(a) each coalition $S_i \in \mathcal{S}_f$ order its buyers in a way to minimize the offloading cost (maximize (17)).
Repeat for every $S_i \in \mathcal{S}_f$
(b) each buyer in a coalition $S_i \in \mathcal{S}_f$ attempts to acquire computation demands in coalition S_i.
Until no peer offloading in the coalition is available.
(c) any buyer who still has unsatisfied computation demand performs SBS-to-cloud offloading.
These two stages are repeated periodically to adapt the partition to environmental changes.

The two coalitions will then decide whether or not to merge. Whenever a Merge decision occurs, a coalition can subsequently investigate the possibility of a Split. Clearly, a Merge or a Split operation is a distributed decision that an SBS (or a coalition of SBSs) can make. After successive Merge-and-Split iterations, the network converges to a partition composed of disjoint coalitions and no coalition has any incentive to further merge or split. In other words, the partition is *Merge-and-Split proof*. The convergence of any Merge-and-Split iterations such as the proposed algorithm is guaranteed as shown in [2]. Upon convergence, the second phase of the actual computation offloading then starts using mechanisms described in Section 3.

4.2 Stability Analysis

The outcome of the above algorithm is a partition of disjoint independent coalitions of SBSs. As an immediate result of the definition of $\mathbb{D}_{hp}$ stability, every partition resulting from proposed algorithm is $\mathbb{D}_{hp}$-stable. In particular, no coalitions of SBSs in the final partition have the incentive to pursue a different coalition formation through Merge or Split. Next, we investigate whether the proposed algorithm can achieve $\mathbb{D}_c$-stability.

A $\mathbb{D}_c$-stable partition has the following properties according to [2]. (i) No SBSs are interested in leaving $\mathcal{S}$ to form other collections in $\mathcal{N}$ (through any operation). (ii) A $\mathbb{D}_c$-stable partition is the *unique* outcome of any *arbitrary* iteration of merge-and-split, if it exists. (iii) A $\mathbb{D}_c$-stable partition $\mathcal{S}$ is a unique $\triangleright$-maximal partition, i.e., for all partition $\mathcal{S}' \neq \mathcal{S}$, we have $\mathcal{S} \triangleright \mathcal{S}'$. Therefore, the $\mathbb{D}_c$-stable partition provides a *Pareto optimal* utility distribution. However, the existence of a $\mathbb{D}_c$ stable partition is not always guaranteed [2]. Nevertheless, we can still have the following result.

Theorem 4.2. *The proposed distributed SBS coalition formation algorithm converges to the Pareto-optimal $\mathbb{D}_c$-stable partition, if such a partition exists. Otherwise, the final partition is $\mathbb{D}_{hp}$-stable.*

Proof. The proof is immediate due to the fact that, when it exist, the $\mathbb{D}_c$-stable partition is a unique outcome of any Merge-and-Split iteration [2], such as any partition resulting from our coalition formation algorithm. □

The stability of the grand coalition (e.g. all SBSs form a single coalition) is of particular interest in the coalitional game theory. It can be easily shown that the considered SBS coalitional game is generally not superadditive and its core is generally empty due to the extra offloading cost (transmission delay, cost and collaboration risk) and hence, the grand coalition is not stable. Instead, independent disjoint coalitions will form. Readers who are interested in more details on the stability of grand coalition in coalitional games are referred to [6, 22].

4.3 Complexity Analysis

The complexity of the proposed coalition formation algorithm lies mainly in the complexity of the Merge and Split operations. For a given network, in one Merge operation, each current coalition attempts to merge with other coalitions in a pairwise manner. In the worst case scenario, every SBS, before finding a suitable merge partner, needs to make a merge attempt with all other SBSs in $\mathcal{N}$. In this case, the first SBS requires $N-1$ attempts for merge, the second requires $N-2$ attempts and so on. The total number of merge attempts in the worst case is thus $\sum_{i=1}^{N}(N-i)$. In practice, the merge process requires a significantly lower number of attempts since finding a suitable partner does not always require to go through all possible merge attempts (once a suitable partner is identified the merge will occur immediately). The complexity is further reduced due to the fact that SBSs do not need to attempt to merge with physically unreachable SBSs. Moreover, after the first run of the algorithm, the initial N non-cooperative SBSs will self-organize into larger coalitions. Subsequent runs of the algorithm will deal with a network composed of a number of coalitions that is much smaller than N.

For the split operation, in the worst case scenario, splitting can involve finding all the possible partitions of the set formed by the SBSs in a single coalition. For a given coalition S, this number is given by the Bell number $\sum_{k=1}^{|S|} \binom{|S|}{k}$ which grows exponentially with the number of SBSs $|S|$ in the coalition. In practice, this split operation is restricted to the formed coalitions, and thus it will be applied to small sets. The split complexity is further reduced due to the fact that, in most scenarios, a coalition does not need to search for all possible split forms. For instance, once a coalition identifies a suitable split structure, the SBSs in this coalition will split, and the search for further split forms is not needed in the current iteration.

5 SIMULATION

Our simulation adopts the widely-used stochastic geometry approach for ultra dense SBS deployment, which is modeled as a homogeneous Poisson Point Process (PPP) [3]. Specifically, we simulate a 100m×200m×50m office building where a set of SBSs are deployed whose locations are chosen according to the PPP with

Table 1: Simulation setup: system parameters

Parameters	Value
Maximum UE transmission power p_m	10 dBm
Maximum SBS transmission power p_s	20 dBm
UE transmission rate requirement r_u	25 Mbps
SBS peer transmission rate requirement r_s	50 Mbps
Frequency (indoor path loss model) f	900 MHz
Indoor path loss exponent v	3.3
Floor penetration loss $L_f(n)$, n=[1,2,3]	[9,12,24] db
SBS density (PPP)	0.15
User density (PPP)	0.6
Task arrival rate of MUEs λ^a	5
System bandwidth W	20 MHz
Cloud offloading delay $d^{c,tx}$	0.3 sec/task

density 0.15. The distribution of MUEs also follows another PPP with density 0.6. The task generation process of each MUE is modeled as a Poisson process with a rate of 5 tasks per time slot. The maximum transmission power of MUEs is set as 10 dBm. The channel model follows the ITU indoor path loss model [24]: $L[\text{db}] = 20\lg f + 10v\lg d_u + L_f(n) - 28$ (The values and corresponding explanations of parameters are shown in TABLE 1). The target MUE-SBS transmission rate is $r_u = 25$ Mbps and hence the corresponding transmission power can be computed. Similar requirements are imposed on the transmissions between SBSs with SBS maximum transmission power 20 dBm and target transmission rate $r_s = 50$ Mbps. Figure 4 shows the SBS deployment and MUE association in one operational time slot used in the simulation.

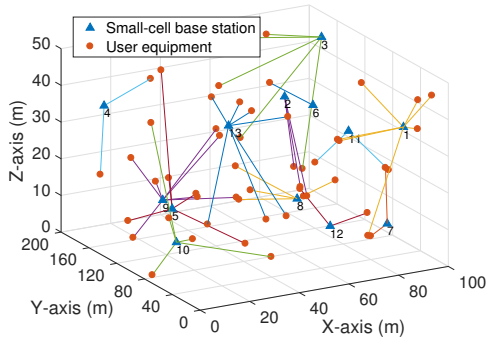


Figure 4: SBS deployment and MUE association

For simplicity, we consider a fixed social trust network throughout the simulation time horizon. Nevertheless, our algorithm is also compatible with evolving trust networks that are updated continuously or periodically according to SBS interactions. The adopted social trust network is illustrated in Figure 5. SBSs that are physically unreachable will not have mutual trust values (e.g. SBS 3 and SBS 10). SBSs that are physical neighbors may also miss mutual trust due to the lack of recent interactions. In this case, a trust value is obtained by finding the shortest path in the social trust network (e.g. SBS 5 → SBS 10 → SBS 13).

The proposed SBS collaboration scheme is compared with three benchmark schemes:

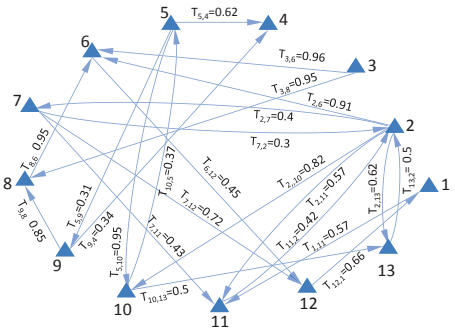


Figure 5: Social trust network

- **Non-cooperative scheme**: Every SBS does not share computation resources with peer SBSs. For overloaded SBSs, all unsatisfied computation tasks will be offloaded to the cloud server.
- **Cloud-offloading minimization**: The scheme greedily exploits the computation resources of peer SBSs to minimize the number of tasks offloaded to the cloud server. Therefore, peer offloading is performed whenever possible regardless of the offloading and risk management costs.
- **Centralized collaborative scheme**: This scheme uses brutal force to search for the best coalition structure that minimizes the total system cost without considering the coalition stability or SBS incentives.

5.1 Coalition formation in one time slot

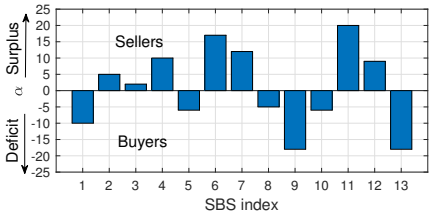


Figure 6: Computation resource surplus/deficit

We exemplify the distributed coalition formation process by considering the computation surplus/deficit profile shown in Figure 6 in one operational time slot. Figure 7 depicts the evolution of the intermediate post-payment utilities $\boldsymbol{\phi}$ during the coalition formation process, where weights in the cost function are set as $w_c = 0.2, w_r = 0.2, w_0 = 1$. The horizontal axis represents the iterations in the Merge-and-Split process and for each iteration, we indicate if it is a Merge operation or a Split operation. Since all SBSs have their own computation tasks to process, they incur positive costs (hence negative utilities). In particular, SBSs 1, 5, 8, 9, 10, 13 have computation resources deficits. Without SBS collaboration, they have to offload the extra computation tasks to the remote cloud, thereby incurring large transmission delay and cloud payment costs. During the coalition formation, each iteration is executed by following the Merge/Split operation that aims to find a Pareto-dominant coalition partition than the current partition. Therefore, after each iteration, at least one of the SBSs improves

its utilities without decreasing the utilities of other SBSs. Figure 8 shows the system utility evolution of each specific Merge or Split operation. We see that the system utility is improved with every Merge/Split operation and after only several iterations, the network converges to a stable partition of coalitions. This indicates that in practice, the complexity of running the proposed algorithm is low and hence, it can be easily implemented.

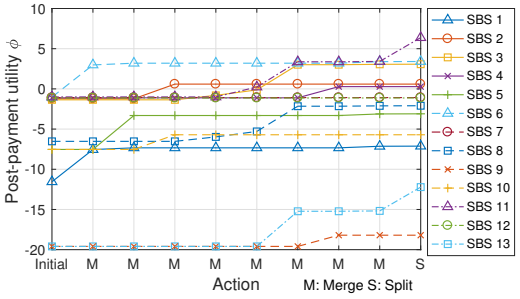


Figure 7: Evolution of post-payment utility ϕ

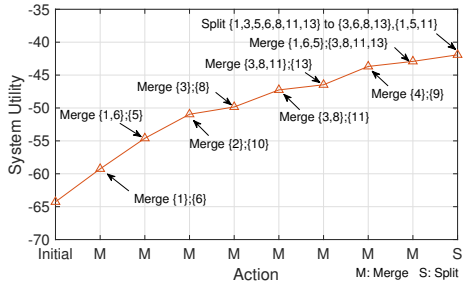


Figure 8: System utility evolution and coalition formation

The final coalitions and the number of exchanged computation tasks are presented in Figure 9, where buyers are matched with nearby sellers. Several points are worth noting. First, a coalition can contain multiple buyers and multiple sellers and is not necessarily just a matching between one buyer and one seller. For example, the coalition $\{4, 9\}$ involves only one buyer SBS 9 and seller SBS 4, whereas the coalition $\{3, 6, 8, 13\}$ involves two buyer SBSs 8, 13 and two seller SBSs 3, 6. In particular, SBS 6 is sharing its computation resources with both SBSs 8 and 13. Second, an SBS may not want to join any coalition. In other words, an SBS may want to form an isolated coalition that contains only itself. In this particular simulation, we observe that SBS 7 and SBS 12 separately form isolated coalitions. As a result, the utilities of these two SBSs stay the same before and after the coalition formation. This can also be observed in Figure 7.

By reading both Figure 6 and Figure 9, we can see that not all computation demands of buyers can be satisfied via SBS coalition. For instance, consider SBS 9, it has a computation resource deficit for 18 tasks, yet it can only offload 10 tasks to SBS 4, which is the only matched seller SBS. In this case, the remaining 8 unsatisfied tasks will be offloaded to the cloud. Figure 10 further shows the computation resource deficit (or surplus) and the actual computation resource bought (or sold) for each SBS. Clearly, the deficit (or surplus) serves an upper bound on what is actually bought (or sold) and hence, the magnitude of the red bar is always larger than that of the corresponding blue bar. In particular, SBSs 7 and 12 do not sell any of their computation resource surplus to other SBSs. In the same figure, we also show the payments made by buyer SBSs and the rewards received by seller SBSs. It can be verified that the total payment equals the total reward within each coalition. For instance, in the coalition $\{3, 6, 8, 13\}$, the total payment is 0.19 + 8.30 made by buyer SBSs 8, 13 and the total reward is 4.13 + 4. 36 received by seller SBSs 3, 6.

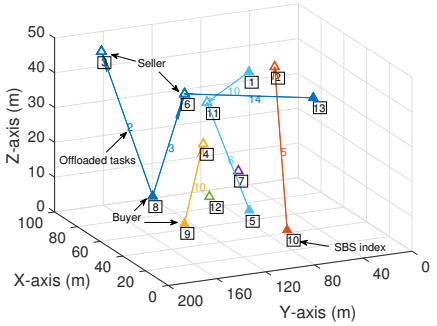


Figure 9: Coalitions and computation peer offloading

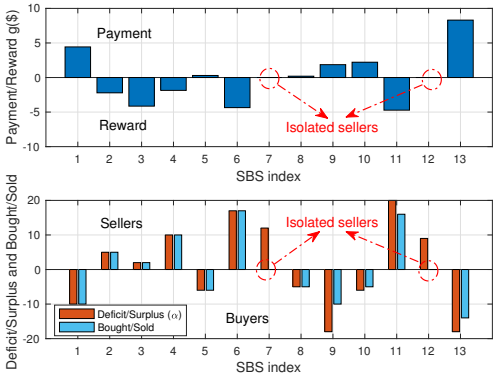


Figure 10: Payment/reward and bought/sold of SBSs

5.2 Performance evaluation and comparison

Figure 11 shows the cost incurred by different parts of the system and offers a comparison with the three benchmarks. (1) The non-cooperative scheme incurs the highest total cost. Since there is no collaboration among the SBSs, each SBS has to offload its extra computation tasks to the remote cloud, thereby incurring a large transmission delay and cloud payment cost. However, since there is no task offloading between SBSs, its operational cost is lowered and totally avoids the risk management cost since all tasks are processed locally or in the secure cloud. (2) The purpose of cloud-offloading minimization scheme is to minimize the number of tasks offloaded to the remote cloud, thereby reducing the transmission delay and cloud payment cost. As can be seen, the cost due to using cloud service is significantly reduced. However, since the optimization ignores the operation cost due to offloading among SBSs

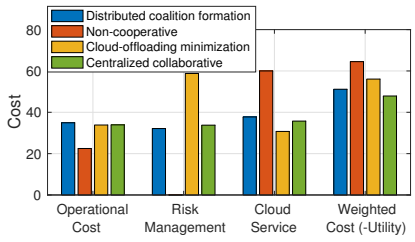


Figure 11: Costs characterization

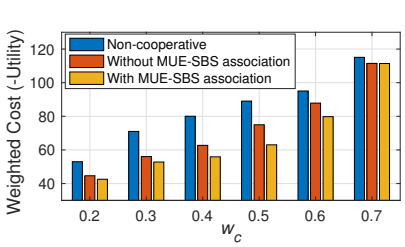


Figure 12: MUE-SBS association scheme

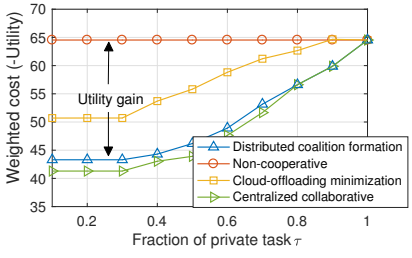


Figure 13: Impact of τ

and especially the risk management cost, the total cost is still high. (3) The centralized collaborative scheme minimizes the overall network cost by jointly considering all cost sources, thereby achieving a much lower total cost. However, the centralized scheme is performed under the assumption that all SBSs are cooperative with no incentive issues, which is not suitable for our considered problem where SBSs are self-interested. (4) Our solution based on the coalitional game gracefully addresses the incentive challenge. As can be seen, the proposed algorithm achieves similar performance to the centralized scheme while ensuring stability of coalitions.

In Figure 12, we demonstrate the benefits achieved by introducing the workload balancing in MUE-SBS association, which is a distinct feature of ultra dense SBS networks and that existing work [26] did not consider. We run the coalition formation algorithm under two settings: with and without MUE-SBS association scheme. As can be observed, the MUE-SBS association scheme helps to reduce the system cost and the cost reduction depends on the value of w_c. When w_c is small, the transmission delay and energy consumption saved by MUE-SBS association scheme are less valued, therefore, a modest cost reduction is realized. As the w_c increases, the edge system benefits more by adopting the MUE-SBS association scheme. However, a too large w_c discourages formation of SBS coalitions which restrains the role of MUE-SBS association.

The performance improvement by SBS coalition also depends on how flexibly computation tasks can be offloaded. Figure 13 shows the impact of the fraction of the private tasks among the total tasks. As can be seen, a larger utility gain is achieved at a lower fraction of private tasks because peer offloading among SBSs is more flexible. Again the proposed coalition formation algorithm significantly reduces the total system cost and achieves close-to-optimal performance.

How much performance improvement (i.e. cost reduction) can be achieved by SBS coalition will depend on the spatial traffic pattern in the network. Intuitively, if all SBSs have a light workload, then there is no need for collaboration, whereas if all SBSs are overloaded, then it is not possible to collaborate. Now, we investigate the performance of collaborative edge computing as a function of the system utilization level (i.e. ratio of expected task number in the network to system computation capacity $\sum_{i\in\mathcal{N}} \omega_i^{\max}$). We simulate a spectrum of the system utilization level by changing the number of MUEs in the network. Figure 14 depicts the impact of the number of MUEs on the collaborative performance in terms of total system cost. When there are only a few MUEs, most of the SBSs can use its own computation resource to satisfy the computation requests. The system is thus a "buyers' market". When there is an excessive number of MUEs, most of the SBSs need extra computation resources. The system is thus a "sellers' market". In both cases, coalitions are difficult to form. The maximum utility gain by collaboration occurs when the system computation capacity matches the number of MUEs. By allowing collaboration among SBSs, the spatial workload intensity heterogeneity is mitigated via workload balancing. In our simulations, the maximum absolute utility gain is achieved at around 50 MUEs and the maximum relative utility gain (41%) is achieved at around 40 MUEs.

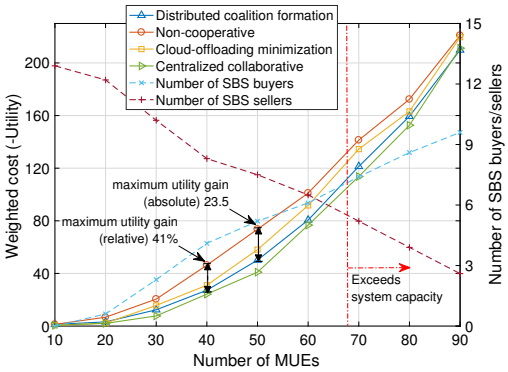


Figure 14: Impact of MUE number

6 RELATED WORK

Resource allocation in small cell networks has been widely studied in the literature. For instance, belief propagation method is applied to manage interference problems in the femtocell network[19]. Orthogonal bandwidth allocation for femtocells is investigated using fractional frequency reuse [11]. However, these works only manage communication resources and do not consider the computing capabilities of SBSs and how SBSs can form a pool of computational resources while providing radio access service. Computational resource sharing is the main concern of geographical load balancing techniques originally proposed for data centers to deal with spatial diversities of workload patterns [12] and electricity prices [13]. However, conventional clouds manage only computational resources without considering the radio access. In edge systems, the provisioning of radio access between the SBSs and end users is a critical design aspect that has a significant impact on the system performance. Recently, with SBSs being considered as a major form of edge devices in the new edge computing paradigm, increasingly many works start to look at the joint optimization of computation

and communication resources of SBSs [4, 15]. However, most of these works assume that the coalitions among the SBSs have already been determined and focus on the offloading decisions of MUEs. In [16, 17], computation clusters of SBSs are optimally built by performing joint allocation of radio and computation resources, focusing on reducing the power consumption and processing complexity. However, the incentive issue of SBSs is not considered.

Since SBSs are typically developed by individual owners, how to provide SBSs with incentives to collaborate is key to improving the overall system performance. In [7, 10], incentive mechanisms based on game theoretic methods are designed to motivate femtocells to adopt open/hybrid radio access mode, which does not consider the cooperative edge computing. In [9], a double auction mechanism is developed for matching user computation request to a set of cloudlets. These works consider incentives of SBSs to provide services to end users. Our paper studies incentives of SBSs to collaborate among each other.

The coalitional game theory is a powerful tool for studying user cooperation problems with individual incentive issues, which has been widely adopted in various problems, e.g. interference management [18], spectrum sensing [21] and smart grids [23]. Recently, it is also introduced in [26] to the collaborative edge computing system for forming Femto-clouds among individual femtocells. Our considered system differs from this paper in the following aspects. First, we consider an ultra dense deployment scenario. Therefore, collaboration does not only occur on the SBS peer offloading level but also on the MUE-SBS association level. Second, we consider socially-trusted collaboration by introducing the social trust network. This allows risk management between SBSs.

7 CONCLUSION

In this paper, we investigated collaborative edge computing in a densely-deployed small cell network, where the SBSs are incentivized to form coalitions and cooperate with SBS peers to increase system utility. Within the coalitions, buyers and sellers are allowed to cooperate at both MUE-SBS association stage and SBS peer offloading stage by exploiting the dense deployment of SBSs. We developed a distributed coalition formation algorithm based on Merge-and-Split rules. The proposed algorithm jointly considers SBS operational cost, cloud service fees, and potential security risks during coalition formation and guarantees the stability of coalitions by following a payment-based incentive mechanism. Our simulation results show that the collaborative edge system can dramatically reduce the system cost by more than 40%. Our study not only provides new guidelines for cooperation among densely deployed edge devices but also delivers important insights for social trust and security issues in edge cooperation, which deserves more future investigation in collaborative edge systems.

REFERENCES

[1] Jeffrey G Andrews, Stefano Buzzi, Wan Choi, Stephen V Hanly, Angel Lozano, Anthony CK Soong, and Jianzhong Charlie Zhang. 2014. What will 5G be? *IEEE Journal on selected areas in communications* 32, 6 (2014), 1065–1082.

[2] Krzysztof R Apt and Andreas Witzel. 2009. A generic approach to coalition formation. *International Game Theory Review* 11, 03 (2009), 347–367.

[3] François Baccelli, Bartlomiej Baszczyszyn, and others. 2010. Stochastic geometry and wireless networks: Volume II Applications. *Foundations and Trends® in Networking* 4, 1–2 (2010), 1–312.

[4] Sergio Barbarossa, Paolo Di Lorenzo, and Stefania Sardellitti. 2014. Computation offloading strategies based on energy minimization under computational rate constraints. In *Networks and Communications (EuCNC), 2014 European Conference on*. IEEE, 1–5.

[5] Tamer Basar and Geert Jan Olsder. 1998. *Dynamic noncooperative game theory*. SIAM.

[6] Anna Bogomolnaia and Matthew O Jackson. 2002. The stability of hedonic coalition structures. *Games and Economic Behavior* 38, 2 (2002), 201–230.

[7] Y. Chen, J. Zhang, and Q. Zhang. 2012. Utility-Aware Refunding Framework for Hybrid Access Femtocell Network. *IEEE Transactions on Wireless Communications* 11, 5 (May 2012), 1688–1697.

[8] Pedro Garcia Lopez, Alberto Montresor, Dick Epema, Anwitaman Datta, Teruo Higashino, Adriana Iamnitchi, Marinho Barcellos, Pascal Felber, and Etienne Riviere. 2015. Edge-centric Computing: Vision and Challenges. *SIGCOMM Comput. Commun. Rev.* 45, 5 (Sept. 2015), 37–42.

[9] A. L. Jin, W. Song, and W. Zhuang. 2016. Auction-Based Resource Allocation for Sharing Cloudlets in Mobile Cloud Computing. *IEEE Transactions on Emerging Topics in Computing* PP, 99 (2016), 1–1.

[10] R. Langar, S. Secci, R. Boutaba, and G. Pujolle. 2015. An Operations Research Game Approach for Resource and Power Allocation in Cooperative Femtocell Networks. *IEEE Transactions on Mobile Computing* 14, 4 (April 2015), 675–687.

[11] J. Y. Lee, S. J. Bae, Y. M. Kwon, and M. Y. Chung. 2011. Interference Analysis for Femtocell Deployment in OFDMA Systems Based on Fractional Frequency Reuse. *IEEE Communications Letters* 15, 4 (April 2011), 425–427. https://doi.org/10.1109/LCOMM.2011.030311.101871

[12] Minghong Lin, Zhenhua Liu, Adam Wierman, and Lachlan LH Andrew. 2012. Online algorithms for geographical load balancing. In *Green Computing Conference (IGCC), 2012 International*. IEEE, 1–10.

[13] J. Luo, L. Rao, and X. Liu. 2015. Spatio-Temporal Load Balancing for Energy Cost Optimization in Distributed Internet Data Centers. *IEEE Transactions on Cloud Computing* 3, 3 (July 2015), 387–397.

[14] Yuyi Mao, Changsheng You, Jun Zhang, Kaibin Huang, and Khaled B Letaief. 2017. Mobile Edge Computing: Survey and Research Outlook. *arXiv preprint arXiv:1701.01090* (2017).

[15] Olga Munoz, Antonio Pascual-Iserte, and Josep Vidal. 2015. Optimization of radio and computational resources for energy efficiency in latency-constrained application offloading. *IEEE Transactions on Vehicular Technology* 64, 10 (2015), 4738–4755.

[16] J. Oueis, E. C. Strinati, and S. Barbarossa. 2014. Small cell clustering for efficient distributed cloud computing. In *2014 IEEE 25th Annual International Symposium on Personal, Indoor, and Mobile Radio Communication (PIMRC)*. 1474–1479.

[17] J. Oueis, E. C. Strinati, and S. Barbarossa. 2015. The Fog Balancing: Load Distribution for Small Cell Cloud Computing. In *2015 IEEE 81st Vehicular Technology Conference (VTC Spring)*. 1–6.

[18] Francesco Pantisano, Mehdi Bennis, Walid Saad, Merouane Debbah, and Matti Latva-Aho. 2013. Interference alignment for cooperative femtocell networks: A game-theoretic approach. *IEEE Transactions on Mobile Computing* 12, 11 (2013), 2233–2246.

[19] S. Rangan and R. Madan. 2012. Belief Propagation Methods for Intercell Interference Coordination in Femtocell Networks. *IEEE Journal on Selected Areas in Communications* 30, 3 (April 2012), 631–640.

[20] J Rivera and R van der Meulen. 2014. Gartner says the Internet of Things will transform the Data Center. *Retrieved August* 5 (2014), 2014.

[21] W. Saad, Z. Han, T. Basar, M. Debbah, and A. Hjorungnes. 2011. Coalition Formation Games for Collaborative Spectrum Sensing. *IEEE Transactions on Vehicular Technology* 60, 1 (Jan 2011), 276–297.

[22] Walid Saad, Zhu Han, Mérouane Debbah, Are Hjorungnes, and Tamer Basar. 2009. Coalitional game theory for communication networks. *IEEE Signal Processing Magazine* 26, 5 (2009), 77–97.

[23] W. Saad, Z. Han, and H. V. Poor. 2011. Coalitional Game Theory for Cooperative Micro-Grid Distribution Networks. In *2011 IEEE International Conference on Communications Workshops (ICC)*. 1–5.

[24] P Series. 2012. Propagation data and prediction methods for the planning of indoor radiocommunication systems and radio local area networks in the frequency range 900 MHz to 100 GHz. *Recommendation ITU-R* (2012), 1238–7.

[25] Weisong Shi, Jie Cao, Quan Zhang, Youhuizi Li, and Lanyu Xu. 2016. Edge computing: Vision and challenges. *IEEE Internet of Things Journal* 3, 5 (2016), 637–646.

[26] S Tanzil, Omid Gharehshiran, and Vikram Krishnamurthy. 2016. A Distributed Coalition Game Approach to Femto-Cloud Formation. *IEEE Transactions on Cloud Computing* (2016).

[27] TROPIC. 2015. Distributed computing storage and radio resource allocation over cooperative femtocells. (2015). http://www.ict-tropic.eu.

[28] Jie Xu, Lixing Chen, and Shaolei Ren. 2017. Online learning for offloading and autoscaling in energy harvesting mobile edge computing. *IEEE Transactions on Cognitive Communications and Networking* PP, P (2017), 1–15.