Computation Peer Offloading for Energy-Constrained Mobile Edge Computing in Small-Cell Networks

Lixing Chen, *Student Member, IEEE*, Sheng Zhou, *Member, IEEE*, and Jie Xu, *Member, IEEE*

***Abstract*— The (ultra-)dense deployment of small-cell base stations (SBSs) endowed with cloud-like computing functionalities paves the way for pervasive mobile edge computing, enabling ultra-low latency and location-awareness for a variety of emerging mobile applications and the Internet of Things. To handle spatially uneven computation workloads in the network, cooperation among SBSs via workload peer offloading is essential to avoid large computation latency at overloaded SBSs and provide high quality of service to end users. However, performing effective peer offloading faces many unique challenges due to limited energy resources committed by self-interested SBS owners, uncertainties in the system dynamics, and co-provisioning of radio access and computing services. This paper develops a novel online SBS peer offloading framework, called online peer offloading (OPEN), by leveraging the Lyapunov technique, in order to maximize the long-term system performance while keeping the energy consumption of SBSs below individual long-term constraints. OPEN works online without requiring information about future system dynamics, yet provides provably near-optimal performance compared with the oracle solution that has the complete future information. In addition, this paper formulates a peer offloading game among SBSs and analyzes its equilibrium and efficiency loss in terms of the price of anarchy to thoroughly understand SBSs' strategic behaviors, thereby enabling decentralized and autonomous peer offloading decision making. Extensive simulations are carried out and show that peer offloading among SBSs dramatically improves the edge computing performance.**

***Index Terms*— Edge computing, load management, energy efficiency, peer-to-peer computing.**

I. Introduction

PERVASIVE mobile devices and the Internet of Things are driving the development of many new applications, turning data and information into actions that create new capabilities, richer experiences and unprecedented economic opportunities. 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

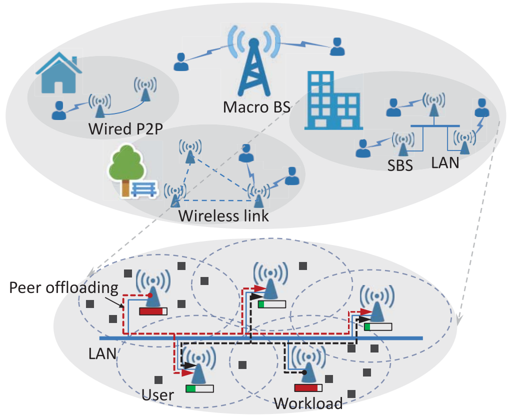


Fig. 1. Illustration of SBS peer offloading.

bandwidth [1]–[3]. 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 [4]. As a remedy to these limitations, mobile edge computing (MEC) [1]–[3] has recently emerged as a new computing paradigm to enable in-situ data processing at the network edge, in close proximity to mobile devices and connected things. Located often just one wireless hop away from the data source, edge computing provides a low-latency offloading infrastructure, and an optimal site for aggregating and analyzing bandwidth-hungry data from end devices.

Considered as a key enabler of MEC, small-cell base stations (SBSs), such as femtocells and picocells, endowed with cloud-like computing and storage capabilities can serve end users' computation requests as a substitute of the cloud [3]. Nonetheless, compared to mega-scale data centers, SBSs are limited in their computing resources. Since the computation workload arrivals in small cell networks can be highly dynamic and heterogeneous, it is very difficult for an individual SBS to provide satisfactory computation service at all times. To overcome these difficulties, cooperation among SBSs can be exploited to enhance MEC performance and improve the efficiency of system resource utilization via computation peer offloading. For instance, a cluster of SBSs can coordinate among themselves to serve mobile users by offloading computation workload from SBSs located in hot spot areas to nearby peer SBSs with light computation workload, thereby balancing workload among the geographically distributed SBSs (see Figure 1 for an illustration). Similar ideas have been investigated for data-center networks to deal with spatial diversities of workload patterns, temperatures, and

Manuscript received March 16, 2017; revised December 6, 2017 and April 25, 2018; accepted May 22, 2018; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor S. Chong. Date of publication June 21, 2018; date of current version August 16, 2018. This work was supported by the Nature Science Foundation of China under Grant 91638204, Grant 61571265, and Grant 61621091. *(Corresponding author: Lixing Chen.)*

L. Chen and J. Xu are with the Department of Electrical and Computer Engineering, University of Miami, Coral Gables, FL 33146 USA (e-mail: lx.chen@miami.edu; jiexu@miami.edu).

S. Zhou is with the Department of Electronic Engineering, Tsinghua University, Beijing 100084, China (e-mail: sheng.zhou@tsinghua.edu.cn).

This paper has supplementary downloadable material available at http://ieeexplore.ieee.org, provided by the authors.

Digital Object Identifier 10.1109/TNET.2018.2841758

electricity prices. In fact, SBS networks are more vulnerable to heterogeneous workload patterns than data center networks which serve an aggregation of computation requests across large physical regions. Since the serving area of each SBS is small, the workload pattern can be affected by many factors such as location, time, and user mobility, therefore becoming very violate and easily leading to uneven workload distribution among the SBSs. Although there have been quite a few works on geographical load balancing in data centers, performing peer offloading in MEC-enabled small cell networks faces unique challenges.

First, small cells are often owned and deployed by individual users. Although incentive mechanism design, which has been widely studied in the literature for systems not limited to small cell networks, plays an important role in incentivizing self-interested users to participate in the collaboration of workload peer offloading, an equally, if not more, important problem is how to maximize the value of the limited resources committed by individual SBS owners. Second, small cells operate in a highly stochastic environment with random workload arrivals in both temporal and spatial domains. As a result, the long-term system performance is more relevant than the immediate performance. However, the limited energy resources committed by the SBS owners make the peer offloading decisions across time intricately intertwined, yet the decisions have to be made without foreseeing the far future. Third, whereas data centers manage only the computing resources, moving the computing resources 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 interplay and interdependency between the management of the two resources under energy constraints.

In this paper, we study computation peer offloading in MEC-enabled small cell networks. Our goal is to maximize the long-term system-wide performance (i.e. minimizing latency) while taking into account the limited energy resources committed by individual SBS owners. The main contributions of this paper are summarized as follows:

1) We develop a novel framework called OPEN (which stands for Online PEer OffloadiNg) for performing stochastic computation peer offloading among a network of MEC-enabled SBSs in an online fashion by leveraging the Lyapunov optimization [5]. We prove that OPEN achieves within a bounded deviation from the optimal system performance that can be achieved by an oracle algorithm that knows the complete future information, while bounding the potential violation of the energy constraints imposed by individual SBS owners.

2) We theoretically characterize the optimal peer offloading strategy. We show that the peer offloading decisions are determined by the *marginal computation cost* (MaCC) – a critical quantity that captures both computation delay cost and energy cost at SBSs. The peer offloading essentially is to evenly distribute MaCCs among SBSs. The SBSs decide their roles (to send or receive workload) based on the pre-offloading MaCCs (i.e., MaCCs before peer offloading). The amount of workload to be offloaded is determined based on optimal post-offloading MaCCs (i.e., MaCCs to achieve after peer offloading) designed by OPEN.

3) We consider both the scenario in which a central entity (e.g. the network operator) collects all current time information and coordinates the peer offloading and the scenario in which SBSs coordinate their peer offloading strategies in a decentralized and autonomous way. For the latter case, we formulate a novel peer offloading game, prove the existence of a Nash equilibrium using the variational inequality technique, and characterize the efficiency loss due to the strategic behaviors of SBSs in terms of the *price of anarchy* (PoA).

4) We run extensive simulations to evaluate the performance of OPEN and verify our analytical results for various system configurations and traffic arrival patterns. The results confirm that our method significantly improves the system performance in terms of latency reduction and energy efficiency.

The rest of this paper is organized as follows. Section II reviews related works. Section III presents the system model and formulates the problem. Section IV develops the OPEN framework and presents the centralized solution for computation peer offloading. Section V formulates and analyzes the peer offloading game. Simulations are carried out in Section VI, followed by the conclusion in Section VII.

II. Related Work

The concept of offloading data and computation in cloud computing is used to address the inherent problems in mobile computing by using resource providers other than the mobile device itself to host the execution of mobile applications [16]. In the most common case, mobile cloud computing means to run an application on a resource rich cloud server located in remote mega-scale data centers, while the mobile device acts like a thin client connecting over to the remote server through 4G/Internet [17]. Recently, the edge computing paradigm [2] (a.k.a. fog computing [18], cloudlet [19], micro datacenter [20]) brings computing resources closer to the end users to enable ultra-low latency and precise location-awareness, thereby supporting a variety of emerging mobile applications such as mobile gaming, augmented reality and autonomous vehicles. Nevertheless, edge servers, such as MEC-enabled SBSs [21], cannot offer the same computation and storage capacities as traditional computing servers.

Many recent works investigate SBS cooperation for improving the system performance, subject to various constraints including local resource availability (e.g. radio resources [6], computational capacities [10], energy consumption budgets [22], and backhaul bandwidth capacity [23]). However, most of these works focus on optimizing the radio access performance only without considering the computing capability of SBSs. In [9] and [10], computation load distribution among the network of SBSs is investigated by considering both radio and computational resource constraints. Clustering algorithms are proposed to maximize users' satisfaction ratio while keeping the communication power consumption low. However, these works focus more on the user-to-SBS offloading side whereas our paper studies the offloading among peer SBSs. More importantly, these works perform myopic optimization whereas our paper studies a problem that is highly coupled across time due to the long-term energy constraints.

TABLE I
COMPARISON WITH EXISTING WORKS

Feature \ Approach	[6], [7]	[8]–[10]	[11]	[12]	[13]	[14], [15]	OPEN (This paper)
Applied stage	UE-to-ES	UE-to-ES	DC-to-DC	DC-to-DC	UE-to-DC	UE-to-ES	ES-to-ES
Radio access aware	Yes	Yes	No	No	No	Yes	Yes
Computation aware	No	Yes	Yes	Yes	Yes	Yes	Yes
System objective	Myopic	Myopic	Long-term	Myopic	Long-term	Myopic	Long-term
Long-term constraints	No	No	Yes (Overall)	No	No	No	Yes (Individual)
Temporal correlation	No	No	Yes	No	Yes	No	Yes
Strategic behavior	No	No	No	No	Yes	Yes	Yes

UE: User equipment; DC: Data Center; ES: Edge Server

Computation workload peer offloading among SBSs is closely related to geographical load balancing techniques originally proposed for data centers to deal with spatial diversities of workload patterns [24], temperatures [25], and electricity prices [26]. Most of these works study load balancing problems that are independent across time [27]. Very few works consider temporally coupled problems. In [24], the temporal dependency is due to the switching costs (turning on/off) of data center servers, which significantly differs from our considered problem. The closest work to our paper is [11], which aims to minimize the long-term operational cost of data centers subject to a long-term water consumption constraint. However, the long-term constraint is imposed on the entire system whereas in our paper each SBS has an individual energy budget constraint. Moreover, we not only provide centralized solutions for peer-offloading but also develop schemes that enable autonomous coordination among SBSs by formulating and studying a peer-offloading game. Several works use reinforcement learning to efficiently manage the resource of geo-distributed data centers [28], [29]. Although the reinforcement learning can also be a potential solution to our peer-offloading problem, there are several challenges for using such a formulation. First, the system states are usually assumed to be Markovian, which may not be true in real systems. Second, large state and action spaces may be needed to capture the various system and decision variables and hence complexity and convergence is a big issue. Third, reinforcement learning often does not capture the long-term energy constraint and may easily violate it. By contrast, our solution is based on the *Lyapunov drift-plus-penalty* framework, which can be applied to more general stochastic systems, does not need to maintain large state and action spaces, and can handle long-term energy constraints.

The formulated peer offloading game is similar to the widely studied congestion game [30] at the first sight. However, there is a crucial difference between these two games: a main assumption in congestion games is that all players have the same cost function for an element; however, in the peer offloading game, cost function of a SBS to retain workload on itself is different from that of other SBSs to offload tasks to that SBS due to the energy consumption concern. This difference demands for new analytical tools for understanding the peer offloading game. For instance, the potential function technique [30] used to establish the existence of a Nash equilibrium in the congestion game does not apply and hence, in this paper, we prove the existence of a Nash equilibrium via the variational inequality technique [31]. Game theoretic modeling was also applied in the MEC computation offloading context in [14] and [15]. These works focus on the computation offloading among multiple UEs to a single BS, which is a different scenario than ours. Table 1 summarizes the differences of proposed strategy from existing works.

III. SYSTEM MODEL

A. Network Model

We consider N SBSs (e.g. femtocells), indexed by $\mathcal{N} = \{1, \dots, N\}$, deployed in a building (residential or enterprise) and connected by the same Local Area Network (LAN). These SBSs are endowed with, albeit limited, edge computing capabilities and hence, User Equipments (UEs) can offload their computation tasks to corresponding serving SBSs via wireless communications for processing. The computing capabilities of SBS i is characterized by its computation service rate f_i (CPU frequency), and the computation service rates of all SBSs in the network are collected by $\boldsymbol{f} = \{f_i\}_{i\in\mathcal{N}}$. Let $\mathcal{M} = \{1, \dots, M\}$ denote the set of all UEs in the building. Each SBS serves a dedicated set of UEs in its serving area, denoted by $\mathcal{M}_i \subseteq \mathcal{M}$. For example, UEs (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. Notice that our algorithm is also compatible with other network structure and association strategies as long as the UE-SBS associations stay unchanged in one peer offloading decision cycle.

B. Workload Arrival Model

The operational timeline is discretized into time slots (e.g. 1–5 minutes) for making peer offloading decisions, which is a much slower time scale than that of task arrivals. In each time slot t, computation tasks originating from UE m is generated according to a Poisson process which is a common assumption on the computation task arrival in edge systems [1]. Let π_m^t denote the rate of the Poisson process for task generation at UE m in time slot t. In each time slot, π_m^t is randomly drawn from $\pi_m^t \in [0, \pi_{\max}]$ to capture the temporal variation in task arrival pattern. Let $\boldsymbol{\pi}^t = \{\pi_m^t\}_{m\in\mathcal{M}}$ denote the task arrival pattern of all UEs in time slot t. The UEs may request for different types of tasks which vary in input data size and required CPU cycles. To simply the system model, we assume that the expected input data size for one task is s (in bits) and the expected number of CPU cycles required by one task is h. The total task arrival rate to SBS i, denoted by ϕ_i^t, is $\phi_i^t = \sum_{m\in\mathcal{M}_i} \pi_m^t$. The task arrival rates to all SBSs are collected in $\boldsymbol{\phi}^t = \{\phi_i^t\}_{i\in\mathcal{N}}$.

C. Transmission Model

1) Transmission Energy Consumption: Transmissions occur on both the wireless link between UEs and SBSs, and the wired link among SBSs. Usually, the energy consumption of wireless transmission dominates and hence we only consider the wireless part. In each time slot t, SBSs serve both uplink and downlink traffic data. We focus on the downlink traffic since transmission energy consumption of SBSs is mainly due to downlink transmission. Suppose each SBS $i \in \mathcal{N}$ operates at a fixed transmission power P_i^d, and the transmissions are operated on orthogonal channels, then the achievable downlink transmission rate $r_{im}^{d,t}$ between UE m and SBS i is given by the Shannon capacity, $r_{im}^{d,t} = W \log_2 \left(1 + \frac{P_i^d H_{im}^t}{\sigma^2}\right)$, where W is the channel bandwidth, H_{im}^t is the channel gain between SBS i and UE m, and σ^2 is the noise power. The downlink traffic consists of the computation result and other communication traffic. Since the size of computation result is usually small, we only consider the communication traffic. Let $w_m^t \in [0, w_{\max}]$ denote the downlink traffic in time slot t, then the energy consumption of SBS i for wireless transmission is

$$E_i^{\text{tx},t} = \sum\nolimits_{m \in \mathcal{M}_i} \frac{P_i^d w_m^t}{r_{im}^{d,t}}. \tag{1}$$

2) UE-to-SBS Transmission Delay: The transmission delay is incurred during UE-to-SBS offloading where UEs send computation tasks to the SBSs through the uplink channel. Let P_m^u be the transmission power of UE m, then the uplink transmission rate between UE m and SBS i, denoted by $r_{im}^{u,t}$, can be also obtained by the Shannon capacity. Therefore, the total transmission delay cost for UEs covered by SBS i is

$$D_i^{u,t} = \sum\nolimits_{m \in \mathcal{M}_i} \frac{s \pi_m^t}{r_{im}^{u,t}}. \tag{2}$$

D. SBS Peer Offloading

Since workload arrivals are often uneven among the SBSs, computation offloading between peer SBSs can be enabled to exploit underused, otherwise wasted, computational resource to improve the overall system efficiency. We assume that tasks can be offloaded only once: if a task is offloaded from SBS i to SBS j, then it will be processed at SBS j and will not be offloaded further or back to SBS i to avoid offloading loops. Let $\boldsymbol{\beta}_{i\cdot}^t = \{\beta_{ij}^t\}_{j \in \mathcal{N}}$ denote the offloading decision of SBS i in time slot t, where β_{ij}^t denotes the fraction of received tasks offloaded from SBS i to SBS j (notice that β_{ii}^t is the fraction that SBS i retains). A peer offloading profile of the whole system is therefore $\boldsymbol{\beta}^t = \{\boldsymbol{\beta}_{i\cdot}^t\}_{i \in \mathcal{N}}$. We further define $\boldsymbol{\beta}_{\cdot i}^t = \{\beta_{ji}^t\}_{j \in \mathcal{N}}$ as the inbound tasks of SBS i, namely the tasks offloaded to SBS i from other SBSs. Clearly, the total workload that will be processed by SBS i is $\omega_i^t(\boldsymbol{\beta}^t) \triangleq \sum_{j \in \mathcal{N}} \beta_{ji}^t$. To better differentiate the two types of workload ϕ_i^t and $\omega_i^t(\boldsymbol{\beta}^t)$, we call ϕ_i^t the *pre-offloading* workload and $\omega_i^t(\boldsymbol{\beta}^t)$ the *post-offloading* workload. A profile $\boldsymbol{\beta}^t$ is feasible if it satisfies: (i) *Positivity*: $\beta_{ij}^t \geq 0$, $\forall i, j \in \mathcal{N}$. The offloaded workload must be non-negative; (ii) *Conservation*: $\sum_{j=1}^N \beta_{ij}^t = \phi_i^t, \forall i \in \mathcal{N}$. The total offloaded workload (including the retained workload) by each SBS must equal its *pre-offloading* workload; (iii) *Stability*: $\omega_i^t(\boldsymbol{\beta}^t) \leq f_i/h$, $\forall i \in \mathcal{N}$. The *post-offloading* workload of each SBS must not exceed its service rate. Let $\mathcal{B}^t$ denote the set of all feasible peer offloading profile.

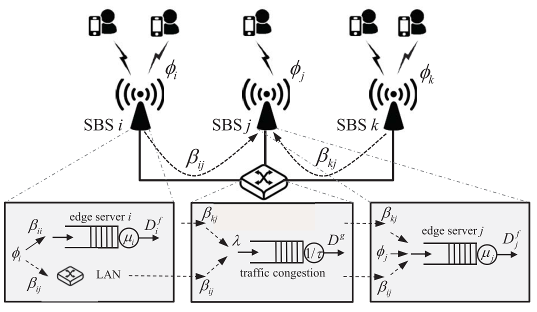


Fig. 2. Illustration of system delay with queuing models. SBS i and SBS k offload workload β_{ij} and β_{kj} to SBS j, respectively. The computation delay model for SBS k is omitted since it is the same as that for SBS i.

Since the LAN bandwidth is limited, peer offloading also causes additional delay due to network congestion. The congestion delay depends on the total traffic in the LAN, denoted by $\lambda^t(\boldsymbol{\beta}^t) = \sum_{i \in \mathcal{N}} \lambda_i^t(\boldsymbol{\beta}^t)$, where $\lambda_i^t(\boldsymbol{\beta}^t) = \sum_{j \in \mathcal{N} \setminus \{i\}} \beta_{ij} = \phi_i^t - \beta_{ii}^t$ is the number of tasks offloaded to other SBSs from SBS i. We assume the data size of computation tasks has an exponential distribution, then the congestion delay $D^{g,t}$ can be modeled as a M/M/1 queuing system [32]:

$$D^{g,t}(\boldsymbol{\beta}^t) = \frac{\tau}{1 - \tau \lambda^t(\boldsymbol{\beta}^t)}, \quad \lambda^t < \frac{1}{\tau}, \tag{3}$$

where τ is the expected delay for sending and receiving s bits (i.e., expected input data size of a computation task) over the LAN without congestion.

E. Computation Model

1) Computation Delay: The computation delay is due to the limited computing capability of SBSs. UEs in the network may request different types of services, therefore the required number of CPU cycles to process a computation task may vary across tasks. We model the distribution of the required number of CPU cycles of individual tasks as an exponential distribution. Given the constant processing rate, the service time of a task therefore follows an exponential distribution. Further considering the Poisson arrival of the computation tasks, the computation delay at each SBS can be modeled as an M/M/1 queuing system [32] and the expected computation delay $D_i^{f,t}$ for one task at SBS i is

$$D_i^{f,t}(\boldsymbol{\beta}^t) = \frac{1}{\mu_i - \omega_i^t(\boldsymbol{\beta}^t)}, \tag{4}$$

where $\mu_i = f_i/h$ is the expected service rate with regard to the number of tasks (i.e. tasks per second) and $\omega_i^t(\boldsymbol{\beta}^t)$ is the workload processed at SBS i given the peer offloading decision $\boldsymbol{\beta}^t$. Figure 2 illustrates the relation between the computation delay and the congestion delay.

2) Computation Energy Consumption: The computation energy consumption at SBS i is load-dependent, denoted as $E_i^{c,t}$. In this paper, we consider a linear computation energy consumption function $E_i^{c,t}(\boldsymbol{\beta}^t) = \kappa \cdot \omega_i^t(\boldsymbol{\beta}^t)$, where $\kappa > 0$ is the energy consumption for executing h (i.e., expected number of CPU cycles required by a computation task) CPU cycles.

F. Problem Formulation

Peer offloading relies on SBSs' cooperative behavior in sharing their computing resources as well as their energy costs. A large body of literature was dedicated to design incentive mechanisms [33], [34] to encourage cooperation among self-interested SBSs (e.g. computing capability, energy budget) to improve the social welfare. The focus of our paper is not to design yet another incentive mechanism. Instead, we design SBS peer offloading strategies taking the SBS committed resources as the input and hence our method can work in conjunction with any existing incentive/cooperation mechanisms. Usually, the decision cycle of the resource scheduling is much longer than that of peer offloading, therefore in this paper we consider each SBS has a predetermined long-term energy consumption constraint as a result of some incentive mechanism.

In each time slot t, the total delay cost of SBS i, defined as a sum of the delays experienced by the tasks arrived at SBSs i, consists of computation delay cost, network congestion delay cost, and UE-to-SBS transmission delay cost:

$$D_i^t(\boldsymbol{\beta}^t) = \sum\nolimits_{j\in\mathcal{N}} \beta_{ij}^t D_j^{f,t}(\boldsymbol{\beta}^t) + \lambda_i^t D^{g,t}(\boldsymbol{\beta}^t) + D_i^{u,t} \tag{5}$$

and the energy consumption of SBS i consists of transmission energy consumption and computation energy consumption:

$$E_i^t(\boldsymbol{\beta}^t) = E_i^{\text{tx},t} + E_i^{c,t}(\boldsymbol{\beta}^t) = E_i^{\text{tx},t} + \kappa\cdot\omega_i^t(\boldsymbol{\beta}^t). \tag{6}$$

The objective of network operator is to minimize the long-term system delay cost given the energy budgets committed by individual SBSs (which are outcomes of the adopted incentive mechanisms). Formally, the problem is

$$\textbf{P1} \quad \min_{\boldsymbol{\beta}^0,\ldots,\boldsymbol{\beta}^{T-1}} \frac{1}{T}\sum\nolimits_{t=0}^{T-1}\sum\nolimits_{i=1}^{N} \mathbb{E}\left\{D_i^t(\boldsymbol{\beta}^t)\right\} \tag{7a}$$

$$\text{s.t. } \frac{1}{T}\sum\nolimits_{t=0}^{T-1}\mathbb{E}\left\{E_i^t(\boldsymbol{\beta}^t)\right\} \le \bar{E}_i, \quad \forall i\in\mathcal{N} \tag{7b}$$

$$E_i^t(\boldsymbol{\beta}^t) \le E_{\max}, \quad \forall i\in\mathcal{N},\ \forall t \tag{7c}$$

$$D_i^t(\boldsymbol{\beta}^t) \le D_{\max}, \quad \forall i\in\mathcal{N},\ \forall t \tag{7d}$$

$$\boldsymbol{\beta}^t \in \mathcal{B}^t, \quad \forall t \tag{7e}$$

Constraint (7b) is the long-term energy budget constraint for each SBS. Constraint (7c) requires that the energy consumption of a SBS does not exceed an upper limit $E_{\max}$ in each time slot. Constraint (7d) indicates that the per-slot delay of each SBS is capped by an upper limit $D_{\max}$ so that the real-time performance is guaranteed in the worst case.

The major challenge that impedes the derivation of optimal solution to **P1** is the lack of future information. Optimally solving **P1** requires complete offline information (task arrivals across all time slots) which is difficult to predict in advance, if not impossible. Moreover, the long-term energy constraints couple the peer offloading decision across different slots: consuming more energy in the current slot will reduce the available energy for future use. These challenges call for an online optimization approach that can efficiently perform peer offloading without foreseeing the future.

Algorithm 1 OPEN

```
Input: control parameter $V$, energy deficit queues
       $\boldsymbol{q}(0) = \boldsymbol{0}$;
Output: offloading decisions $\boldsymbol{\beta}^0,\ldots,\boldsymbol{\beta}^{T-1}$;
1 for $t = 0$ to $T-1$ do
2     Observe workload arrival $\boldsymbol{\phi}^t$ and feasible peer
      offloading strategy set $\mathcal{B}^t$ ;
3     Solving P2 to get optimal $\boldsymbol{\beta}^t$ in time slot $t$:
      $\min_{\boldsymbol{\beta}^t\in\mathcal{B}^t} \sum_{i\in\mathcal{N}} (V\cdot D_i^t(\boldsymbol{\beta}^t) + q_i(t)\cdot E_i^t(\boldsymbol{\beta}^t))$ ;
4     Update the deficit for all SBS $i$:
5     $q_i(t+1) = [q_i(t) + E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i]^+$
6 end
7 return $\boldsymbol{\beta}^1,\ldots,\boldsymbol{\beta}^{T-1}$;
```

IV. Online SBS Peer Offloading

In this section, we develop a novel framework for making online SBS peer offloading decisions, called OPEN (Online SBS PEer offloadiNg) by leveraging the Lyapunov technique. OPEN converts **P1** to per-slot optimization problems solvable with only current information. We consider both the case in which the network operator coordinates the SBS peer offloading in a centralized way (this section) and the case in which SBSs make peer offloading decisions among themselves in an autonomous manner (next section).

A. Lyapunov Optimization Based Online Algorithm

In the optimization problem **P1**, the long-term energy constraints of SBSs couple the peer offloading decisions across times slots. To address this challenge, we leverage the *Lyapunov drift-plus-penalty technique* [5] and construct a (virtual) energy deficit queue for each SBS to guide the peer offloading decisions to follow the long-term energy constraints. We define a set of energy deficit queues $\boldsymbol{q}(t) = \{q_i(t)\}_{i\in\mathcal{N}}$, one for each SBS, and let $q_i(0) = 0, \forall i \in \mathcal{N}$. For each SBS $i \in \mathcal{N}$, its energy deficit queue evolves as follows:

$$q_i(t+1) = \max\{q_i(t) + E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i, 0\}, \tag{8}$$

where $q_i(t)$ is the queue length in time slot t, indicating the deviation of current energy consumption from the long-term energy constraint of SBS i.

Next, we present the online algorithm OPEN (Algorithm 1) for solving **P1**. In OPEN, the network operator determines the peer offloading strategy in each time slot t by solving the optimization problem **P2**, as presented below:

$$\textbf{P2} \quad \min_{\boldsymbol{\beta}^t\in\mathcal{B}^t} \sum\nolimits_{i=1}^{N}\left(V\cdot D_i^t(\boldsymbol{\beta}^t) + q_i(t)\cdot E_i^t(\boldsymbol{\beta}^t)\right)$$
$$\text{s.t. (7c),} \quad \text{(7d) and (7e)}$$

The objective in **P2** is designed based on *Lyapunov drift-plus-penalty* framework. The rationale behind this design will be explained later in Section IV-C. The first term in **P2** is to minimize the system delay and the second term is added aiming to satisfy the long-term energy constraint (7b) in an online manner; the positive control parameter V is used to adjust the trade-off between these two purposes.

To give a brief explanation, by considering the additional term $\sum_{i=1}^{N} q_i(t)E_i^t(\boldsymbol{\beta}^t)$, the network operator takes into account the energy deficits of SBSs in current-slot decision making: when $\boldsymbol{q}(t)$ is larger, minimizing the energy deficits is more critical for network operator. Thus, OPEN works following the philosophy of "if violate the energy budget, then use less energy", and hence the long-term energy constraint can be satisfied in the long run without foreseeing the future information. Later in this section, we will rigorously prove the performance of OPEN in terms of system delay cost and long-term energy consumption. Now, to complete OPEN, it remains to solve the optimization problem **P2**. Notice that solving **P2** requires only currently available information as input.

B. Centralized Solution to OPEN

In this subsection, we consider the existence of a centralized controller who collects the complete current-slot information from all SBSs, solves the per-slot problem **P2**, and coordinates SBS peer offloading in each time slot t. Before proceeding to the solution, we rewrite the objective function of **P2** as below:

$$\begin{aligned}
&\sum\nolimits_{i\in\mathcal{N}} \left(VD_i^t(\boldsymbol{\beta}^t) + q_i(t)E_i^t(\boldsymbol{\beta}^t)\right)\\
&= \sum\nolimits_{i\in\mathcal{N}} V\Big(\sum\nolimits_{j\in\mathcal{N}} \beta_{ij}^t D_j^{f,t}(\boldsymbol{\beta}^t) + \lambda_i^t(\boldsymbol{\beta}^t)D^{g,t}(\boldsymbol{\beta}^t) + D_i^{u,t}\Big)\\
&\quad + \sum\nolimits_{i\in\mathcal{N}} q_i(t)(E_i^{\text{tx},t} + E_i^{c,t}(\boldsymbol{\beta}^t))\\
&= \underbrace{\sum\nolimits_{i\in\mathcal{N}} V\left(\frac{V\omega_i^t(\boldsymbol{\beta}^t)}{\mu_i - \omega_i^t(\boldsymbol{\beta}^t)} + \kappa q_i(t)\omega_i^t(\boldsymbol{\beta}^t)\right) + \frac{V\tau\lambda^t(\boldsymbol{\beta}^t)}{1-\tau\lambda^t(\boldsymbol{\beta}^t)}}_{\text{decision-dependent}}\\
&\quad + \underbrace{\sum\nolimits_{i\in\mathcal{N}} \left(VD_i^{u,t} + q_iE_i^{\text{tx},t}\right)}_{\text{decision-independent}}. \qquad (9)
\end{aligned}$$

The objective function of **P2** can be divided into two parts: (i) a decision-dependent part which is a weighted sum of the computation delay cost, the computation energy consumption, and the network congestion delay cost; (ii) a decision-independent part which relates to the UE-to-SBS transmission delay and SBS-to-UE energy consumption. Therefore, we focus on the decision-dependent part for solving **P2**. Although the decision-independent part does not affect the solution of **P2** directly, its second term (i.e., $E^{\text{tx},t}$) will affect the energy deficit queue updating and hence indirectly affects peer offloading decisions in the long-run.

Notice that **P2** is solved in each time slot, for ease of exposition, we drop the time index for variables. Moreover, instead of optimizing $\boldsymbol{\beta}^t$ directly, we alternatively optimize the amount of workload $\omega_i^t(\boldsymbol{\beta})$ each SBS should accommodate, and the corresponding total traffic in the LAN $\lambda^t(\boldsymbol{\beta})$. By rewriting $\omega_i^t(\boldsymbol{\beta})$ as ω_i and $\lambda^t(\boldsymbol{\beta})$ as λ, **P2** is therefore equivalent to:

$$\textbf{P2-S} \quad \min_{\omega_i\in\Omega_i,\lambda\in\Lambda} \sum_{i\in\mathcal{N}} \left(\frac{V\omega_i}{\mu_i-\omega_i} + \kappa q_i\omega_i\right) + \frac{V\tau\lambda}{1-\tau\lambda} \qquad (10a)$$

$$\text{s.t. } E_i(\omega_i,\lambda) \le E_{\max}, \quad \forall i\in\mathcal{N} \qquad (10b)$$

$$D_i(\omega_i,\lambda) \le D_{\max}, \quad \forall i\in\mathcal{N} \qquad (10c)$$

$$\omega_i \in \Omega_i, \quad \lambda\in\Lambda, \ \forall i\in\mathcal{N} \qquad (10d)$$

where Ω_i in (10d) is the feasible space for ω_i determined by the mapping $\omega_i : \mathcal{B} \to \Omega_i$, and similarly Λ is determined by $\lambda : \mathcal{B} \to \Lambda$. Notice that, although ω_is and λ are written as independent variables, they are deterministic functions of a particular $\boldsymbol{\beta}$ in each time slot. To capture the relation between ω_is and λ, we introduce and closely follow a workload flow equation when solving **P2-S**, which will be shown shortly.

Next, we give the optimal solution for the above optimization problem starting with classifying SBSs into the following three categories:

1) Source SBS ($\mathcal{R}$). A SBS is a source SBS if it offloads a positive portion of its *pre-offloading* workloads to other SBSs and processes the rest of workloads locally. Moreover, it does not receive any workload from other SBSs ($0 \le \omega_i < \phi_i$).

2) Neutral SBS ($\mathcal{U}$): A SBS is a neutral SBS if it processes all its *pre-offloading* workloads locally and does not receive any workload from other SBSs ($\omega_i = \phi_i$).

3) Sink SBS ($\mathcal{S}$): A SBS is a sink SBS if it receives workloads from other SBSs and does not offload workload to others ($\omega_i > \phi_i$).

Notice that in our categorization, there is no SBS such that it offloads workloads to other SBSs while receiving workloads from other SBSs. This is because it can be easily shown that having such SBSs result in suboptimal solutions to **P2** due to the extra network congestion delay. To assist the presentation of the optimal solution, we define two auxiliary functions.

Definition 1: Define $d_i(\omega_i) \triangleq \frac{\partial}{\partial\omega_i}[\omega_i D_i^f(\omega_i)] = \frac{\mu_i}{(\mu_i-\omega_i)^2}$ as the marginal computation delay function for SBS $i, \forall i\in\mathcal{N}$; $g(\lambda) \triangleq \frac{\partial}{\partial\lambda}[\lambda D^g(\lambda)] = \frac{\tau}{(1-\tau\lambda)^2}$ as the marginal congestion delay function.

Specifically, $d_i(\omega_i)$ is the marginal value of the computation delay function when ω_i tasks are processed at SBS i; and $g(\lambda)$ is the marginal value of the congestion delay function with $s\cdot\lambda$ bits traffic in the LAN.

We define $\xi_i \triangleq Vd_i(\phi_i)+\kappa q_i$ as the *pre-offloading* Marginal Computation Cost (MaCC), taking into account both the computation delay cost and the computation energy consumption if SBS i processes all its tasks locally. Based on ξ_i, Theorem 1 shows the optimal SBS categorization, workload allocation, and corresponding traffic in LAN.

Theorem 1: The category that SBS i belongs to, the optimal post-offloading workload ω_i^*, and the corresponding traffic in LAN λ^* can be determined based on pre-offloading MaCC ξ_i and a parameter α:

(a) If $\xi_i < \alpha$, then $i\in\mathcal{S}$ and $\omega_i^* = d_i^{-1}(\frac{1}{V}(\alpha - \kappa q_i))$;

(b) If $\alpha \le \xi_i \le \alpha + Vg(\lambda^*)$, then $i\in\mathcal{U}$ and $\omega_i^* = \phi_i$;

(c) If $\xi_i > \alpha + Vg(\lambda^*)$, then $i\in\mathcal{R}$ and $\omega_i^* = [d_i^{-1}(\frac{1}{V}(\alpha + Vg(\lambda^*) - \kappa q_i))]^+$; where λ^*, α are the solution to the workload flow equation:

$$\underbrace{\sum\nolimits_{i\in\mathcal{S}} \left(d_i^{-1}(\frac{1}{V}(\alpha-\kappa q_i)) - \phi_i\right)}_{\lambda^S\text{: inbound workloads to sinks}}$$

$$= \underbrace{\sum\nolimits_{i\in\mathcal{R}} \left(\phi_i - [d_i^{-1}(\frac{1}{V}(\alpha + Vg(\lambda^*) - \kappa q_i))]^+\right)}_{\lambda^R\text{: outbound workloads from sources}}. \qquad (11)$$

Proof: See Appendix A in supplementary file. □

In Theorem 1, α is a Lagrange multiplier that equals the unique optimal *post-offloading* MaCC (i.e. $Vd_i(\omega_i^*) + \kappa q_i$) of sink SBSs. Part ($a$) indicates that SBSs with *pre-offloading*

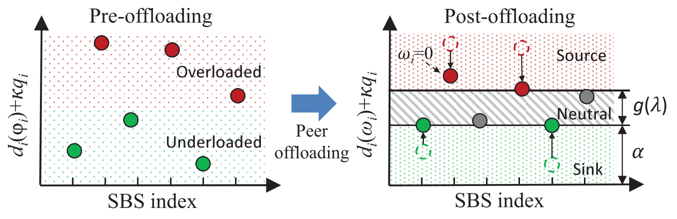


Fig. 3. SBS categorization and changes in the marginal computation costs.

Algorithm 2 OPEN-Centralized

```
Input: SBS services rates: $\mu_1, \mu_2, \ldots, \mu_N$; SBS
       workload arrival rates: $\phi_1, \phi_2, \ldots, \phi_N$; Network
       mean communication time: $\tau$.
1  Initialization: $\omega_i \leftarrow \phi_i, i \in \mathcal{N}$;
2  Get pre-offloading MaCC for SBSs $\xi_i = Vd_i(\phi_i) + \kappa q_i$;
3  Find $\xi_{\max} = \max_{i\in\mathcal{N}} \xi_i$ and $\xi_{\min} = \min_{i\in\mathcal{N}} \xi_i$;
4  If $\xi_{\min} + Vg(0) \geq \xi_{\max}$, then STOP (no peer offloading
   is required);
5  $a \leftarrow \xi_{\min}$; $b \leftarrow \xi_{\max}$;
6  while $|\lambda^S(\alpha) - \lambda^R(\alpha)| \geq \tilde{\epsilon}$ do
7      $\lambda^S(\alpha) \leftarrow 0$, $\lambda^R(\alpha) \leftarrow 0$;
8      $\alpha \leftarrow \frac{1}{2}(a+b)$;
9      Get in order: $\mathcal{S}(\alpha)$, $\lambda^S(\alpha)$, $\mathcal{R}(\alpha)$, $\mathcal{U}(\alpha)$, $\lambda^R(\alpha)$ ;
10     if $\lambda^S(\alpha) > \lambda^R(\alpha)$ then $b \leftarrow \alpha$;
11     else $a \leftarrow \alpha$;
12 end
13 Determine $\omega_i^*$ and $\lambda^*$ based on Theorem 1;
```

MaCC less than α will serve as sink SBSs and their post-offloading MaCCs will be equal to α. Part (b) implies that the *pre-offloading* MaCC of a neutral SBS is no less than α but no larger than the sum of α and the marginal congestion delay cost. This means that no other SBSs would benefit from offloading workloads to neutral SBSs and at the same time neutral SBS receive no benefits by performing peer offloading. For a source SBS, its *pre-offloading* MaCC is larger than the sum of α and the marginal congestion delay cost and therefore, it tends to offload workloads to other SBSs until its *post-offloading* MaCC reduces to $\alpha + Vg(\lambda^*)$ (i.e. $\omega_i^* = d_i^{-1}(V^{-1}(\alpha + Vg(\lambda^*) - \kappa q_i))$ or no more workload can be further offloaded (i.e. $\omega_i^* = 0$). Figure 3 depicts the categorization of SBSs and the difference between their *pre-offloading* and *post-offloading* MaCCs.

Since directly deriving a solution for α is impossible, we develop an iterative algorithm called OPEN-Centralized for solving **P2**. OPEN-Centralized uses binary search to obtain α under the workload flow equation, which is summarized in Algorithm 2. In each iteration, the algorithm first determines a set of sink SBSs ($\mathcal{S}$) according to the parameter α, then the corresponding amount of inbound workload is determined. Given the total workload $\lambda = \lambda^S$ transmitted in the LAN, the algorithm determines the source SBSs ($\mathcal{R}$), neutral SBSs ($\mathcal{U}$) and then calculates the outbound workload λ^R. If λ^R equals λ^S, then the optimal α is found; otherwise, the algorithm updates α and goes into the next iteration. Notice that the algorithm outputs the optimal workload allocation of SBSs, $\omega_i^*, \forall i \in \mathcal{N}$ and the corresponding traffic in the LAN λ^*. Any peer offloading strategy that realizes the optimal workload allocation is an optimal peer offloading profile $\boldsymbol{\beta}^*$ for **P2**.

C. Performance Analysis of OPEN

In this section, we rigorously prove the performance of OPEN. It is shown in [5] that, with the designed energy deficit queue $q_i(t)$, the long-term energy constraint (7b) for each SBS is enforced if the queues $q_i(t)$ are stable, i.e. $\lim_{T\to\infty} \mathbb{E}\{q_i(t)\}/T = 0$. We define a *quadratic Lyapunov function* $L(\boldsymbol{q}(t))$ as: $L(\boldsymbol{q}(t)) \triangleq \frac{1}{2}\sum_{i=1}^N q_i^2(t)$. It represents a scalar metric of the queue length in all virtual queues. A small value of $L(\boldsymbol{q}(t))$ implies that all the queue backlogs are small, which means the virtual queues have strong stability. To keep the virtual queues stable (i.e., to enforce the energy constraints) by persistently pushing the Lyapunov function towards a lower value, we introduce *one-slot Lyapunov drift* $\Delta(\boldsymbol{q}(t))$:

$$\begin{aligned}
&\Delta(\boldsymbol{q}(t))\\
&\triangleq \mathbb{E}\left\{L(\boldsymbol{q}(t+1)) - L(\boldsymbol{q}(t))|\boldsymbol{q}(t)\right\}\\
&= \frac{1}{2}\sum\nolimits_{i=1}^N \mathbb{E}\left\{q_i^2(t+1) - q_i^2(t)|\boldsymbol{q}(t)\right\}\\
&\overset{(\dagger)}{\leq} \frac{1}{2}\sum\nolimits_{i=1}^N \mathbb{E}\left\{(q_i(t) + E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i)^2 - q_i^2(t)|\boldsymbol{q}(t)\right\}\\
&\leq \frac{1}{2}\sum\nolimits_{i=1}^N (E_{\max} - \bar{E}_i)^2 + \sum\nolimits_{i=1}^N q_i(t)\mathbb{E}\left\{E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i|\boldsymbol{q}(t)\right\}
\end{aligned}$$

The inequality (†) comes from $(q_i(t) + E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i)^2 \geq [\max(q_i(t) + E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i, 0)]^2$. By extending the Lyapunov drift to an optimization problem, our objective is to minimize a supremum bound on the following *drift-plus-penalty* expression in each time slot:

$$\begin{aligned}
&\Delta(\boldsymbol{q}(t)) + V\sum_{i=1}^N \mathbb{E}\left\{D_i^t(\boldsymbol{\beta}^t)|\boldsymbol{q}(t)\right\}\\
&\quad \leq V\sum_{i=1}^N \mathbb{E}\left\{D_i^t(\boldsymbol{\beta}^t)|\boldsymbol{q}(t)\right\}\\
&\qquad + B + \sum\nolimits_{i=1}^N q_i(t)\mathbb{E}\left\{(E_i^t(\boldsymbol{\beta}^t) - \bar{E}_i)|\boldsymbol{q}(t)\right\}, \quad (12)
\end{aligned}$$

where $B = \frac{1}{2}\sum_{i=1}^N (E_{\max} - \bar{E}_i)^2$. Notice that OPEN (Line 3 in Algorithm 1) exactly minimizes the right hand side of (12). The parameter $V \geq 0$ controls the *delay-energy deficit* tradeoff, i.e., how much we shall emphasize the delay minimization compared to the energy deficit. Next we give a rigorous performance bound of OPEN compared to the optimal solution to **P1**.

Theorem 2: Following the optimal peer offloading decision $\boldsymbol{\beta}^{*,t}$ obtained by OPEN-Centralized, the long-term system delay cost satisfies:

$$\lim_{T\to\infty} \frac{1}{T}\sum\nolimits_{t=0}^{T-1}\sum\nolimits_{i=1}^N \mathbb{E}\left\{D_i^t(\boldsymbol{\beta}^{*,t})\right\} < D_{sys}^{opt} + \frac{B}{V}, \quad (13)$$

and the long-term energy deficit of SBSs satisfies:

$$\begin{aligned}
&\lim_{T\to\infty} \frac{1}{T}\sum\nolimits_{t=0}^{T-1}\sum\nolimits_{i=1}^N \mathbb{E}\left\{E_i^t(\boldsymbol{\beta}^{*,t}) - \bar{E}_i\right\}\\
&\qquad\qquad \leq \frac{1}{\epsilon}\left(B + V(D_{sys}^{\max} - D_{sys}^{opt})\right), \quad (14)
\end{aligned}$$

where $D_{sys}^{opt} = \lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N} \mathbb{E}\{D_i^t(\boldsymbol{\beta}^{opt,t})\}$ is the optimal system delay to **P1***, $D_{sys}^{\max} = ND_{\max}$ is the largest system delay cost, and $\epsilon > 0$ is a constant which represents the long-term energy surplus achieved by some stationary strategy.*

Proof: See Appendix B in supplementary file. □

The above theorem demonstrates an $[O(1/V), O(V)]$ *delay-energy deficit tradeoff*. OPEN asymptotically achieves the optimal performance of the offline problem **P1** by letting $V \to \infty$. However, the optimal system delay cost is achieved at the price of a larger energy deficit, as a larger deficit queue is required to stabilize the system and hence postpones the convergence. The long-term energy deficit bound in (14) implies that the time-average energy deficit grows linearly with V. Notice that the total energy consumption of the overall system stays almost the same regardless of the peer offloading decision. This is because all computation tasks are accommodated within the edge system and the same amount of energy will be spent to process these tasks (assuming the energy due to wired transmission is negligible). The real issue is where these tasks are processed and how much energy each SBS should spend given its long-term energy constraint.

V. Autonomous SBS Peer Offloading

In the previous section, the network operator coordinates SBS peer offloading in a centralized way. However, the small cell network is often a distributed system where there is no central authority controlling the workload allocation. Moreover, individual SBSs may not have the complete information of the system, which impedes the derivation of social optimal solution. In this section, we formulate OPEN as a non-cooperative game where SBSs minimize their own costs in a decentralized and autonomous way. We analyze the existence of Nash Equilibrium (NE) and the efficiency loss due to decentralized coordination compared to the centralized coordination in terms of the *Price of Anarchy* (PoA).

A. Game Formulation

We first define a non-cooperative game $\Gamma \triangleq (\mathcal{N}, \{\mathcal{B}_{i\cdot}\}_{i\in\mathcal{N}}, \{K_i\}_{i\in\mathcal{N}})$, where $\mathcal{N}$ is the set of SBSs, $\mathcal{B}_{i\cdot}$ is the set of feasible peer offloading strategies for SBS i, and K_i is the cost function for each SBS i defined as $K_i(\boldsymbol{\beta}^t) = VD_i^t(\boldsymbol{\beta}^t) + q_i(t)E_i^t(\boldsymbol{\beta}^t)$. In the autonomous scenario, each SBS aims to minimize its own cost by adjusting its own peer offloading strategy in each time slot t. Without causing confusions, we also drop the time index t in this section. The Nash equilibrium of this game is defined as follows.

Definition 2 (Nash Equilibrium): A Nash equilibrium of the SBS peer offloading game defined above is a peer offloading profile $\boldsymbol{\beta}^{NE} = \{\boldsymbol{\beta}_{i\cdot}^{NE}\}_{i\in\mathcal{N}}$ such that for every SBS $i \in \mathcal{N}$:

$$\boldsymbol{\beta}_{i\cdot}^{NE} \in \arg\min_{\tilde{\boldsymbol{\beta}}_{i\cdot}} K_i(\boldsymbol{\beta}_{1\cdot}^{NE}, \boldsymbol{\beta}_{2\cdot}^{NE}, \ldots, \tilde{\boldsymbol{\beta}}_{i\cdot}, \ldots, \boldsymbol{\beta}_{N\cdot}^{NE}). \quad (15)$$

At the Nash equilibrium, a SBS cannot further decrease its cost by unilaterally choosing a different peer offloading strategy when the strategies of the other SBSs are fixed. The equilibrium peer offloading profile can be found when each SBS's strategy is a *best response* to the other SBSs' strategies. Before proceeding with the analysis, we give an equivalent

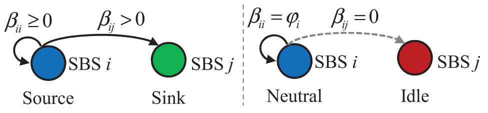


Fig. 4. Illustration of SBS categories with respect to SBS i.

expression of $K_i(\boldsymbol{\beta})$ by considering only the part that depends on SBS i's peer offloading strategy $\boldsymbol{\beta}_{i\cdot}$:

$$\begin{aligned} &C_i(\boldsymbol{\beta}_{i\cdot}) \\ &= \underbrace{\beta_{ii}\left(\frac{V}{\mu_i - \omega_i(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})} + \kappa q_i\right)}_{\text{cost due to local processing}} \\ &\quad + \underbrace{\sum\nolimits_{j\in\mathcal{N}\setminus\{i\}} \beta_{ij}\left(\frac{V}{\mu_j - \omega_j(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})} + \frac{V\tau}{1-\tau\lambda(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})}\right)}_{\text{cost due to peer offloading}} \end{aligned} \quad (16)$$

where $\boldsymbol{\beta}_{-i\cdot}$ is the peer offloading decisions of other SBSs except SBS i. The first part of (16) is the cost incurred by processing the retained workload locally (i.e. the computation delay cost of itself and the energy consumption) and the second part is the cost incurred by performing peer offloading (i.e. the computation delay cost on other SBSs and the network congestion delay cost). To facilitate our analysis, (16) can be further represented as:

$$C_i(\boldsymbol{\beta}_{i\cdot}) = \sum\nolimits_{j\in\mathcal{N}} \beta_{ij} c_{ij}(\boldsymbol{\beta}_{i\cdot}). \quad (17)$$

where $c_{ij}(\boldsymbol{\beta}_{i\cdot})$,

$$= \begin{cases} \dfrac{V}{\mu_j - \omega_j(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})} + \dfrac{V\tau}{1-\tau\lambda(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})}, & \text{if } j \neq i \\ \dfrac{V}{\mu_i - \omega_i(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot})} + \kappa q_i, & \text{if } j = i \end{cases}$$

Then, in the autonomous SBS peer offloading, the best response problem for each SBS $i \in \mathcal{N}$ becomes:

$$\begin{aligned} \textbf{P3} \min_{\boldsymbol{\beta}_{i\cdot}\in\mathcal{B}_{i\cdot}} &\sum\nolimits_{j\in\mathcal{N}} \beta_{ij} c_{ij}(\boldsymbol{\beta}_{i\cdot}) && (18a) \\ \text{s.t. } & E_i(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot}) \leq E_{\max} && (18b) \\ & D_i(\boldsymbol{\beta}_{i\cdot}, \boldsymbol{\beta}_{-i\cdot}) \leq D_{\max} && (18c) \\ & \boldsymbol{\beta}_{i\cdot} \in \mathcal{B}_{i\cdot} && (18d) \end{aligned}$$

The following theorem establishes the existence of a Nash equilibrium in the SBS peer offloading game.

Theorem 3 (Existence of Nash Equilibrium): The SBS peer offloading game admits at least one Nash equilibrium.

Proof: See Appendix C in supplementary file. □

B. Algorithm for Achieving Nash Equilibrium

In this subsection, we first present the *best-response* algorithm which is used to obtain the *best response* strategy $\boldsymbol{\beta}_{i\cdot}^{\mathrm{BR}}$ for each SBS. Then all SBSs take turns in a round-robin fashion to perform the best-response algorithm until a Nash equilibrium $\boldsymbol{\beta}^{\mathrm{NE}}$ is reached.

Analogous to the previous section, SBSs are classified into different categories (see illustration in Fig.4):

1) Source SBS ($\mathcal{R}$). A SBS is a source SBS if it offloads a positive portion of its *pre-offloading* workloads to other SBSs and processes the rest of workloads locally ($0 \leq \beta_{ii} < \phi_i$).

2) Neutral SBS ($\mathcal{U}$). A SBS is a neutral SBS if it processes all its *pre-offloading* workloads locally ($\beta_{ii} = \phi_i$).

3) Idle SBS with respect to SBS i ($\mathcal{I}_i$). A SBS is an idle SBS with respect to SBS i if it does not receive any workload from SBS i ($\beta_{ij} = 0$).

4) Sink SBS with respect to SBS i ($\mathcal{S}_i$). A SBS is a sink SBS with respect to SBS i if it receives workload from SBS i ($\beta_{ij} > 0$).

The first two categories are defined depending on how the SBS handles its own *pre-offloading* workloads. The last two categories are defined depending on how the SBS handles other SBSs' workloads. Since there are $N-1$ other SBSs, the categories are defined with respect to each SBS. Unlike the categories in Section IV, these categories are not mutually exclusive and hence, a SBS can belong to multiple categories at the same time.

To solve the best-response problem in (18a) for each SBS i, we define two auxiliary functions.

Definition 3: Define $d_{ij}(\beta_{ij}) \triangleq \frac{\partial[\beta_{ij} D^f_j(\beta_{ij})]}{\partial \beta_{ij}} = \frac{\mu_{ij}}{(\mu_{ij}-\beta_{ij})^2}$ *as the pair-specific marginal computation delay function;* $g_i(\lambda_i) \triangleq \frac{\partial[\lambda_i D^g(\lambda_i)]}{\partial \lambda_i} = \frac{\tau \Lambda_{-i}}{(\Lambda_{-i}-\tau\lambda_i)^2}$ *as the pair-specific marginal congestion delay function, where* $\mu_{ij} = \mu_j - \sum_{k=1,k\neq i}^N \beta_{kj}$ *and* $\Lambda_{-i} = 1 - \tau \sum_{k=1,k\neq i}^N \lambda_k$.

The *pre-offloading* Pair-specific Marginal Computation Cost (PMaCC) is therefore

$$\xi_{ij} = \begin{cases} V d_{ij}(0), & j \neq i \\ V d_{ii}(\phi_i) + \kappa q_i, & j = i, \end{cases} \tag{19}$$

by initializing $\beta_{ii} = \phi_i$ and $\beta_{ij} = 0, \forall j \in \mathcal{N}, j \neq i$.

Theorem 4: In the SBS peer offloading game, the category that SBS i belongs to and its best response peer offloading strategy $\boldsymbol{\beta}_{i\cdot}^{BR}$ *can be decided as follow:*

For SBS i itself:

(a) If $\xi_{ii} > \alpha_i + V g_i(\lambda_i^{BR})$, *then* $i \in \mathcal{R}$ *and* $\beta_{ii}^{BR} = \left[d_{ii}^{-1}\left(\frac{1}{V}(\alpha_i + V g_i(\lambda_i^{BR}) - \kappa q_i)\right)\right]^+$;

(b) If $\xi_{ii} \leq \alpha_i + V g_i(\lambda_i^{BR})$, *then* $i \in \mathcal{U}$ *and* $\beta_{ii}^{BR} = \phi_i$;

For SBS j other than i ($j \neq i$):

(c) If $\xi_{ij} > \alpha_i$, *then* $j \in \mathcal{I}_i$ *and* $\beta_{ij}^{BR} = 0$;

(d) If $\xi_{ij} < \alpha_i$, *then* $j \in \mathcal{S}_i$ *and* $\beta_{ij}^{BR} = d_{ij}^{-1}(\frac{\alpha_i}{V})$;

where λ^{BR}, α_i *are the solution to workload flow equation*

$$\underbrace{\sum_{j\in\mathcal{S}_i} d_{ij}^{-1}(\frac{\alpha_i}{V})}_{\lambda_i^S\text{: inbound workload to } \mathcal{S}_i} = \mathbf{1}\{i \in \mathcal{R}\} \cdot \underbrace{\left(\phi_i - [d_{ij}^{-1}(\frac{1}{V}(\alpha_i + V g_i(\lambda_i^{BR}) - q_i\kappa))]^+\right)}_{\lambda_i^R\text{: outbound workload from SBS } i}. \tag{20}$$

Proof: See Appendix D in supplementary file. □

Theorem 4 can be explained in a similar way as Theorem 1. Figure 5 depicts the changes of marginal computation costs when SBS i performs best response. One major difference is that in the best-response algorithm, SBS i determines its sink SBSs by examining only the marginal computation delay cost (i.e., d_{ij}), regardless of the marginal energy consumption cost (κq_j) of other SBSs. This is actually intuitive since each SBS aims to minimize its own cost rather than the overall system cost. The solution for α_i can be obtained by a binary search under the workload flow equation (20) as designed by the best-response algorithm in Algorithm 3.

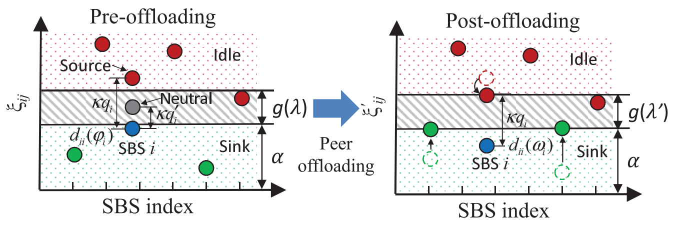


Fig. 5. Changes in the marginal computation costs after best response.

Algorithm 3 Best-Response

```
Input: $\phi_i$, $V$, $q_i(t)$, $\boldsymbol{\beta}_{-i}$.
Output: optimal peer offloading strategy $\boldsymbol{\beta}_{i\cdot}^*$.
1  $\beta_{ij} \leftarrow 0 (j \neq i), \beta_{ii} \leftarrow \phi_i$;
2  Calculate $\xi_{ij}, \forall j \in \mathcal{N}$ as in (19);
3  Find $\xi_{i,\max} = \max_{j\in\mathcal{N},j\neq i} \xi_{ij}$; $\xi_{i,\min} = \min_{j\in\mathcal{N},j\neq i} \xi_{ij}$;
4  If $\xi_{i,\min} + V g_i(0) > \xi_{ii}$ STOP (no peer offloading is
   required);
5  $a \leftarrow \xi_{i,\min}$;
6  $b \leftarrow \xi_{i,\max}$;
7  while $|\lambda_i^S(\alpha_i) - \lambda_i^R(\alpha_i)| \geq \tilde{\epsilon}$ do
8      $\lambda_i^S(\alpha_i) \leftarrow 0$, $\lambda_i^R(\alpha_i) \leftarrow 0$;
9      $\alpha_j \leftarrow \frac{1}{2}(a+b)$;
10     Get in order: $\mathcal{S}_i(\alpha_i)$, $\lambda_i^S(\alpha_i)$, $\mathcal{R}(\alpha_i)$, $\mathcal{U}(\alpha_i)$, $\lambda_i^R(\alpha_i)$,
       $\boldsymbol{\beta}_{i\cdot}$ according to Theorem 4;
11     if $\lambda_i^S(\alpha_i) > \lambda_i^R(\alpha_i)$ then $b \leftarrow \alpha_i$;
12     else $a \leftarrow \alpha_i$;
13 end
14 return $\boldsymbol{\beta}_{i\cdot}^{\text{BR}}$;
```

With the best-response algorithm, we propose the algorithm OPEN-Autonomous to obtain the Nash equilibrium $\boldsymbol{\beta}^{\text{NE}}$, where SBSs take turns to run the best-response algorithm in a round-robin fashion. The iteration terminates if the total cost change of SBSs is less than a sufficiently small tolerance, in which case the SBS sends a terminating message to be propagated among SBSs. An important question is whether such best-response based algorithm indeed converges. There exists results about the convergence of such algorithm in the context of routing in parallel links [35]. For our peer offloading game there exists a unique Nash equilibrium because the cost function of the players are continuous, convex and increasing. Our simulation of in Section VI also confirms the convergence of the best-response algorithm.

C. Price of Anarchy

We now analyze the price of anarchy (PoA) of the SBS peer offloading game, which is a measure of efficiency loss due to the strategic behavior of players. Recall that $\boldsymbol{\beta}^*$ is the centralized optimal solution that minimizes the system-wide cost in each time slot. Let $\mathcal{B}^{\text{NE}}$ be the set of Nash equilibria of SBS peer offloading game. The PoA is defined as:

$$\text{PoA} \triangleq \frac{\max_{\boldsymbol{\beta}\in\mathcal{B}^{\text{NE}}} C(\boldsymbol{\beta})}{C(\boldsymbol{\beta}^*)} = \frac{\max_{\boldsymbol{\beta}\in\mathcal{B}^{\text{NE}}} \sum_{i\in\mathcal{N}} C_i(\boldsymbol{\beta}_{i\cdot})}{\sum_{i\in\mathcal{N}} C_i(\boldsymbol{\beta}^*_{i\cdot})}.$$

In the following, we give a general bound on the PoA.

*Theorem 5 (Bound of PoA): Let $\boldsymbol{\beta}^{NE}$ be a Nash equilibrium and $\boldsymbol{\beta}^*$ be the optimal peer offloading profile for the per-slot problem **P2**. Then the PoA of the non-cooperative SBS peer offloading game satisfies:*

$$1 \le \text{PoA} \le \frac{1}{1-\rho(\boldsymbol{c})},$$

where $\rho(\boldsymbol{c}) \triangleq \sup_{j\in\mathcal{N}} \rho(\boldsymbol{c}_{\cdot j})$, and each $\rho(\boldsymbol{c}_{\cdot j})$ is bounded by

$$\rho(\boldsymbol{c}_{\cdot j}) \le \frac{\eta(\boldsymbol{c}_{\cdot j})}{3+\frac{4}{\delta(\boldsymbol{c}_{\cdot j})(N-1)}}, \quad \forall j\in\mathcal{N}.$$

where $\delta(\boldsymbol{c}_{\cdot j}) = \sup\limits_{i,k\in\mathcal{N}} \frac{c'_{ij}(\boldsymbol{\beta}_{i\cdot})}{c'_{kj}(\boldsymbol{\beta}_{k\cdot})}$, $\eta(\boldsymbol{c}_{\cdot j}) = \sup\limits_{i,k\in\mathcal{N}} \frac{\omega_j c'_{ij}(\boldsymbol{\beta}_{i\cdot})}{c_{kj}(\boldsymbol{\beta}_{k\cdot})}$, $\forall \boldsymbol{\beta}_{i\cdot}\in\mathcal{B}_{i\cdot}, \forall\boldsymbol{\beta}_{k\cdot}\in\mathcal{B}_{k\cdot}$, and $c'_{ij}(\boldsymbol{\beta}_{i\cdot}) = \frac{\partial c_{ij}(\boldsymbol{\beta}_{i\cdot})}{\partial \beta_{ij}}$.

Proof: See Appendix E in supplementary file. □

According to Theorem 5, we see that the bound of PoA is mainly decided by $\delta(\boldsymbol{c}_{\cdot j}) = \sup_{i,k\in\mathcal{N}} \frac{c'_{ij}(\boldsymbol{\beta}_{i\cdot})}{c'_{kj}(\boldsymbol{\beta}_{k\cdot})}$, i.e., the maximal ratio of SBSs' pair-specific marginal cost values with respect to SBS j. A larger $\delta(\boldsymbol{c}_{\cdot j})$ leads to a larger $\rho(\boldsymbol{c}_{\cdot j})$, consequently, a larger PoA. The $\delta(\boldsymbol{c}_{\cdot j})$ grows with the heterogeneity levels in task arrival rates and computation capacity among SBSs. For example, suppose we have a SBS with an extremely large service rate, which gives a very large $\delta(\boldsymbol{c}_{\cdot j})$. Then, all other SBSs tend to offload workload to that SBS, which causes a large congestion delay in the LAN and hence increases the PoA value. Theoretically quantifying PoA value is hard since it heavily depends on the task arrival pattern and system configuration. In the simulation, we measure the PoA value in each time slot.

We have proved that for each time slot t the PoA of the peer offloading game is bounded. Let $\varrho^{\max}$ be the maximum achievable PoA across all time slots. Then, we have:

$$\begin{aligned}&\sum_{i=1}^{N} \left(V\cdot D_i^t(\boldsymbol{\beta}^{\text{NE},t}) + q_i(t)\cdot E_i^t(\boldsymbol{\beta}^{\text{NE},t})\right)\\ &\quad\le \varrho^{\max}\sum_{i=1}^{N}\left(V\cdot D_i^t(\boldsymbol{\beta}^{*,t}) + q_i(t)\cdot E_i^t(\boldsymbol{\beta}^{*,t})\right). \quad (21)\end{aligned}$$

The strategic behavior of SBSs in the peer offloading game cause performance loss in both delay and energy efficiency. The following theorem rigorously shows the performance guarantee of OPEN-Autonomous.

Theorem 6: Following the peer offloading decision $\boldsymbol{\beta}^{NE,t}$ obtained by OPEN-Autonomous, the long-term system delay cost satisfies:

$$\begin{aligned}&\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\mathbb{E}\left\{D_i^t(\boldsymbol{\beta}^{NE,t})\right\}\\ &\quad\le \varrho^{\max}\Psi\left(\frac{\varrho^{\max}-1}{\varrho^{\max}}\bar{E}^{\max}\right)+\frac{B}{V},\end{aligned}$$

and the long-term energy deficit of SBSs satisfies:

$$\begin{aligned}&\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N}\mathbb{E}\left\{E_i^t(\boldsymbol{\beta}^{NE,t})-\bar{E}_i\right\}\\ &\quad\le\frac{B+V(\varrho^{\max}D_{sys}^{\max}-D_{sys}^{opt})}{\varrho^{\max}\epsilon-(\varrho^{\max}-1)\bar{E}^{\max}},\end{aligned}$$

where $\varrho^{\max}$ is the largest PoA *value; $\Psi(\frac{\varrho^{\max}-1}{\varrho^{\max}}\bar{E}^{\max})$ is the delay performance achieved by stationary policy L satisfying energy consumption $\mathbb{E}\{E_i(\boldsymbol{\beta}^{L,t})-\bar{E}_i\} \le -\frac{\varrho^{\max}-1}{\varrho^{\max}}\bar{E}^{\max}$, where $\bar{E}^{\max} = \max_{i\in\mathcal{N}}\bar{E}_i$; $\epsilon > 0$ is a constant which represents the long-term energy surplus achieved by some stationary strategy.*

Proof: See Appendix F in supplementary file. □

Theorem 6 still shows a $[O(V), O(1/V)]$ tradeoff between delay and energy deficit. However, the delay performance achieved is no longer comparable with the optimal system delay $D_{\text{sys}}^{\text{opt}}$. Instead, the bound of delay performance is defined on the stationary policies with a reduced long-term energy constrain $\lim_{T\to\infty}\frac{1}{T}\sum_{t=0}^{T}\mathbb{E}\{E_i(\boldsymbol{\beta}^t)\} \le \bar{E}^{\max}/\varrho^{\max}$. Notice that if $\varrho^{\max}=1$, then Theorem 6 is identical to Theorem 2.

VI. Simulation

Systematic simulations are carried out to evaluate the performance of proposed algorithm under various system settings. We assume that the edge network is deployed in a commercial complex where the business tenants deploy their own SBSs and edge servers to serve their employees. The scale of the considered commercial complex should be large (>100,000ft^2 [36]), such that multiple SBSs are likely to be deployed by different business tenants and the collaboration among SBSs can be exploited. We simulated a 100m×100m commercial complex (107,639ft^2) served by a set of SBSs whose locations are decided by homogeneous Poisson Point Process (PPP). The main advantage of PPP is that it captures the fact that SBSs are randomly deployed by individual owners. The density of PPP process is set as 10^{-3}. With this PPP density and commercial complex area, the expected number of SBSs generated by PPP is 10 which is also the average number of business tenants in a commercial complex within an area around 100,000ft^2 [37]. Here, we assume that on average each business tenant will deploy an SBS. In each time slot, the UEs are randomly scattered in the network. Since the average working space of a worker is 250 square feet [38], the expected number of users in the commercial complex is 400. Considering the variation in the occupancy rate across the time, we assume that the number of UEs in each time slot is randomly drawn from $[200, 600]$. Each UE is randomly assigned to one of its nearby SBSs. For an arbitrary UE, its task generation follows a Poisson process with arrival rate $\pi_m^t \in [0,4]$task/sec. The expected number of CPU cycles for each task is $h = 40$M. Consider that the energy consumption of one CPU cycles is 8.2nJ, the expected energy consumption for each task at SBS n is $\kappa_n = 9\times10^{-5}$Wh and the long-term energy constraint is set as 22W·h per hour. The expected input data size of each task is $s = 0.2$Mb. Therefore, for a typical 100Mb fast Ethernet LAN, the expected transmission delay for one task is $\tau = 200$ms. The channel gain H_m^t

TABLE II

SIMULATION SETUP: SYSTEM PARAMETERS

Parameters	Value
Task arrival rate from UE m, π_m^t	$[0, 4]$task/sec
Expected num. of CPU cycles per-task, h	40M
CPU frequency of edge server at SBS n, f_n	3GHz
Expected input data size per-task, s	0.2Mb
Transmission delay for one task in LAN, τ	200ms
Wireless transmission frequency, f^{TX}	900MHz
Distance power loss coefficient, N_L	20
Wireless channel bandwidth, W	20MHz
Noise power, δ^2	-174dBm/Hz
Transmission power of UEs, P_m^u	10dBm
Expected energy consumption per-task, κ_n	9×10^{-5}W·h
Long-term energy constraint, $\bar{E}_n$	22W·h (per hour)

for calculating the wireless transmission is modeled by the indoor path-loss: $L[\text{dB}] = 20\log(f^{\text{TX}}[\text{MHz}]) + N_L \log(d[\text{m}]) - 28$. Other important parameters are listed in Table II. The performances of OPEN-Centralized (OPEN-C) and OPEN-Autonomous (OPEN-A) are compared with three benchmarks:

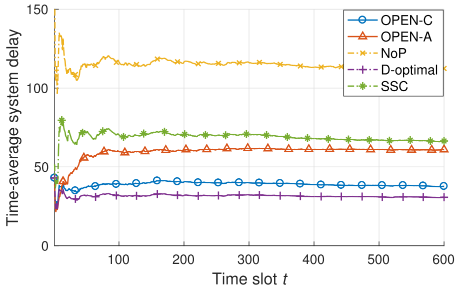
1) No Peer offloading (NoP): peer offloading among SBSs is not enabled in the network. Each SBS processes all the tasks received from the end users. Moreover, the long-term constraint is not enforced since some SBSs have to exceed energy constraint to satisfy all the tasks due to the heterogeneity in spatial task arrival.

2) Delay-Optimal (D-Optimal): we apply the method in [39] where SBS peer offloading is considered as a static load balancing problem aiming to achieve the lowest system delay regardless of the long-term energy constraints.

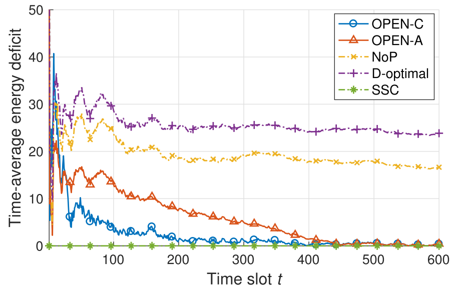
3) Single-Slot Constraint (SSC): Instead of following a long-term energy constraint, the network operator poses a hard energy constraint in each time slot, i.e. $E_i^t(\boldsymbol{\beta}^t) \le \bar{E}_i$, such that the long-term energy constraint is satisfied.

A. Run-Time Performance Evaluation

Figure 6 shows the long-term system performance obtained by running OPEN and we mainly focus on two metrics: the time-average system delay cost in Figure 6(a) and the time-average energy deficit in Figure 6(b). It can be observed that without peer offloading the edge system bears a high delay cost and a large energy deficit, since SBSs can be easily overloaded due to spatially and temporally heterogeneous task arrival pattern. By contrast, other three schemes with peer offloading enabled (D-Optimal, SSC, and OPEN) achieve much lower system delay cost. Specifically, D-Optimal achieves the lowest delay cost since it is designed to minimize the delay cost by fully utilizing the computation resource regardless of the energy constraints. Therefore, D-Optimal incurs a large amount of energy deficit as shown in Figure 6(b). The main purpose of OPEN is to follow the long-term energy constraint of each SBS while minimizing the system delay. As can be observed in Figure 6(b), the time-average energy deficits of both OPEN-C and OPEN-A coverage to zero, which means that the long-term energy constraints are satisfied by running OPEN. Moreover, OPEN-C achieves a close-to-optimal delay cost and OPEN-A incurs a slightly higher delay cost due to the strategic behaviors of selfish SBSs. The SSC scheme poses an energy constraint in each time slot in order to satisfy the long-term energy constraints. As a result, the energy deficit of



(a)



(b)

Fig. 6. Comparison of time-average performances. (a) Time average system delay. (b) Time average energy deficit.

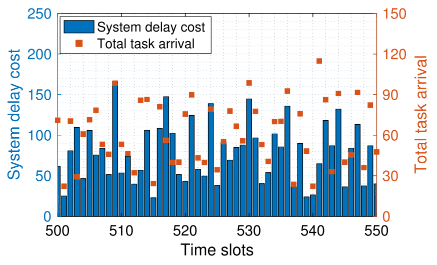


Fig. 7. System dynamics (system delay cost).

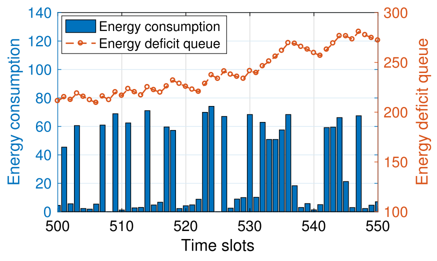


Fig. 8. System dynamics (energy consumption).

SSC is zero across all the time slots. However, SSC makes the energy scheduling less flexible and therefore does not handle well the heterogeneity of temporal task arrival pattern and results in a large system delay cost.

B. System Dynamics

Figure 7 and Figure 8 show the system delay cost and the energy consumption from the 500th to 550th time slot, respectively. We see that the system delay cost is mainly decided by the total task arrival rate in the network which varies across the time slots. Usually, a larger task arrival rate will result in a higher system delay cost. Figure 8 depicts the energy consumption of one particular SBS and

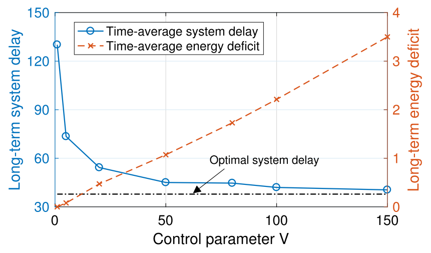


Fig. 9. Impact of control parameter V.

the corresponding energy deficit queue in each time slot to exemplify how the energy deficit queue works to guide the energy usage. For example, from the 530th to 535th time slot, the SBS uses a large amount of energy and enlarges the energy deficit. Therefore, in the following 5 time slots, OPEN reduces the energy consumption to cut the energy deficit. In this way, the long-term energy constraints of SBSs can be satisfied.

C. Impact of Control Parameter V

Figure 9 shows the impact of control parameter V on the performance of OPEN. The result presents a $[O(1/V), O(V)]$ trade-off between the long-term system delay cost and the long-term energy deficit, which is consistent with our theoretical analysis. With a larger V, OPEN emphasizes more on the system delay cost and is less concerned with the energy deficit. As V grows to the infinity, OPEN is able to achieve the optimal delay cost. It is hard to define an optimal value for V since a lower system delay cost is achieved at the cost of larger energy deficit. However, it still offers a guideline for picking an appropriate V. In this particular simulation, the network operator is recommended to choose, for example, $V = 50$ for two reasons: (i) OPEN has already achieved close-to-optimal delay and little improvement is available by increasing V; (ii) the energy deficit is much smaller compared to the energy deficit achieving the optimal delay.

D. Composition of System Delay

Figure 10 and Figure 11 depict the composition of system delay cost for OPEN-C and OPEN-A, receptively. For OPEN-C, the computation delay cost takes up a large proportion of the system delay cost and the congestion delay cost is relatively small. By contrast, the congestion delay cost becomes the main part of the system delay cost in OPEN-A. The non-cooperative behavior of SBSs results in a large volume of traffic exchange in the LAN as SBSs tend to offload workload to other SBSs in order to save their own energy consumption. Note that the communication delay cost is independent of peer offloading and hence it is the same for OPEN-C and OPEN-A.

E. Price of Anarchy

Since it is difficult to quantify theoretically the upper bound of PoA, we measure PoA values in the simulation. Figure 12 depicts the objective value (**P2**) achieved by OPEN-C and OPEN-A from the 50th time slot to the 100th time slot from a simulation run of a total of 600 time slots. It is clearly shown that in each time slot OPEN-C achieves a strictly smaller

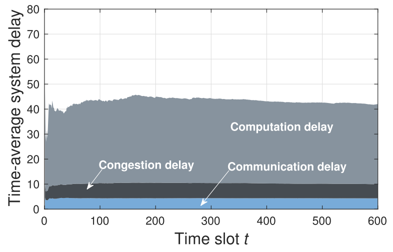


Fig. 10. Delay cost composition of OPEN-C.

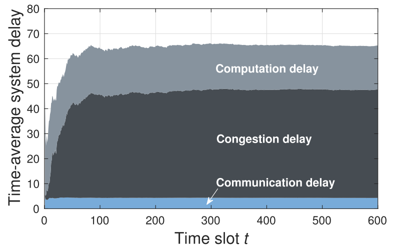


Fig. 11. Delay cost composition of OPEN-A.

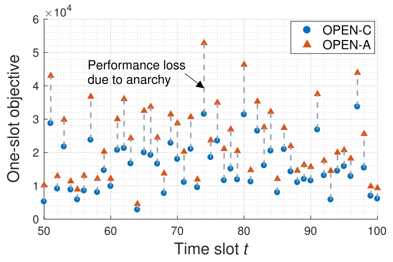


Fig. 12. Price of anarchy.

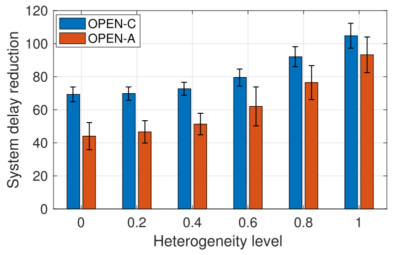


Fig. 13. Impact of system heterogeneity.

value than OPEN-A, which means that OPEN-C outperforms OPEN-A in every time slot. For the entire 600 time slots, the mean PoA value is 1.54 and the maximum PoA value is 2.42.

F. System Heterogeneity

Figure 13 shows the impact of heterogeneity on the performance of OPEN. In particular, the heterogeneity of spatial task arrival pattern is considered: we regularly split the entire network into 4×4 grids and define an expected task arrival rate $\pi_i^{\text{grid}} \sim \mathcal{N}(10, \sigma_s^2), i = 1, 2, \dots, 16$ for each grid which is drawn from a normal distribution with σ_s being the standard deviation. The task arrival rate of UE m is set as π_i^{grid} if UE m belongs to grid i. The level of heterogeneity is varied by changing the standard deviation σ_s, which is normalized with respect to a maximum value $\sigma_{s,\max}$ as $\sigma_s/\sigma_{s,\max}$. The result in Figure 13 shows that for both OPEN-C and OPEN-A, a better system performance in terms of reduced

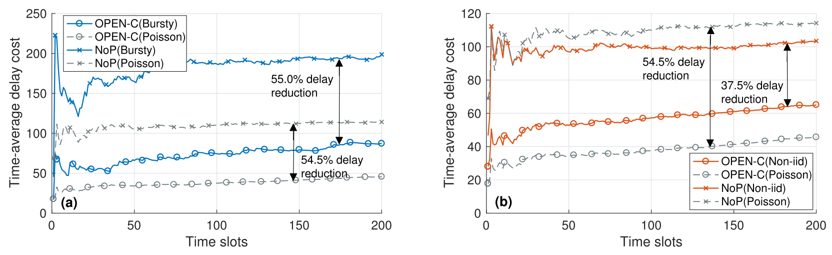


Fig. 14. Impact of task arrival realization: (a) bursty arrival, (b) non-i.i.d. arrival.

TABLE III
ANALYSIS OF ALGORITHM RUNTIME

Algorithm	OPEN-C	OPEN-A
Algorithm runtime	0.83 $\pm$ 0.57 ms	28.1 $\pm$ 4.84 ms
Information exchange	$\approx$ 0.2 ms	$\approx$ 8 ms
Computation delay (per task)	1.25 $\pm$ 0.38 sec	2.07 $\pm$ 0.65 sec
Peer offloading decision cycle	1 min	1 min

delay cost is achieved with a higher heterogeneity level. This is because (some) SBSs are more likely to be overloaded with a higher heterogeneity level and therefore OPEN can better help to reduce the system delay by balancing the workload.

G. Time Complexity

Since it also takes time for OPEN to derive peer offloading decisions, especially for OPEN-A, we measured the runtime of OPEN on a typical PC to see its practical overhead. The simulation is run on a DELL PRECISION T3600 workstation with Intel(R) Xeon(R) CPU 2.8GHz and the results are presented in TABLE III. OPEN-C incurs an extremely low overhead, taking only 0.83ms on average (with standard deviation 0.57ms) to derive solution. By contrast, OPEN-A needs much longer time, namely 28.1ms, to obtain the Nash equilibrium since the SBSs have to take turns to run the best-response algorithm. However, the runtime for both OPEN-C and OPEN-A is negligible compared to the 1-minute peer offloading decision cycle (i.e. duration of one time slot). In addition, the information exchange is necessary to run OPEN: for OPEN-C, the centralized controller collects the information from each SBS at the beginning of each decision cycle; for OPEN-A, SBSs need to exchange the peer offloading decisions immediately after executing the best-response algorithm. TABLE III also shows the estimation of delay incurred by information exchange. Assume the bandwidth of the LAN is 100Mbps and the size of a message packet is 800 Bytes, the delays of information exchange incurred by OPEN-C and OPEN-A are approximately 0.2ms and 8ms, which are also negligible compared to the peer offloading decision cycle.

H. Impact of Task Arrival Realization

Notice that the task arrival pattern in the real-world system may not follow the assumed Poisson process. To analyze the practicality of OPEN, we implement it with different task arrival realizations. Fig. 14 shows the performances achieved by OPEN and NoP with two task arrival realizations: bursty arrival and non-i.i.d. arrival. We can see from Fig.14(a) that OPEN and NoP both have a higher delay cost in the bursty arrival case compared to that in the Poisson arrival case. This is because the tasks are more likely to queue up at edge servers when bursts occur. However, we see that the proposed algorithm still provides a 55.0% delay reduction with bursty arrival which is similar to that of Poisson arrival. In the non-i.i.d. case, we use a Markov process to model a task arrival pattern where the intervals of task arrivals are determined by certain transition probabilities. We see from Fig. 14(b) that the delay reduction achieved by OPEN in the non-i.i.d case slightly decreases compared to that in the Poisson arrival case. However, applying OPEN still offers an obvious delay reduction, 37.5%, for the edge system. These two examples indicate that the proposed algorithm can offer considerable performance improvement for the edge system even if the real task arrival does not closely follow the Poisson process.

VII. CONCLUSION

In this paper, we investigated peer offloading schemes in MEC-enabled small cell networks where heterogeneous task arrival pattern in both spatial and temporal domains is considered. We developed OPEN, a novel online peer offloading framework to optimize the edge computing performance under limited energy resources committed by individual SBSs without requiring information on future system dynamics. The proposed framework allows both centralized and autonomous decision making, and provide provable performance guarantee. We also showed that OPEN incurs acceptable overhead in practice. However, there are still issues that need to be addressed. In the current system model, we considered a simple structure of LAN to model the congestion delay during peer offloading. How to extend our analysis to more sophisticated and practical congestion scenarios needs further investigation. Moreover, the performance guarantee of OPEN rests on the assumption that the observations of task arrival rates in the current slot are precise, which may not hold for all network systems. Therefore, future efforts are needed to take into consideration of imprecise estimation of task arrival.

REFERENCES

[1] Y. Mao, C. You, J. Zhang, K. Huang, and K. B. Letaief, "A survey on mobile edge computing: The communication perspective," *IEEE Commun. Surveys Tuts.*, vol. 19, no. 4, pp. 2322–2358, 4th Quart., 2017.

[2] W. Shi, J. Cao, Q. Zhang, Y. Li, and L. Xu, "Edge computing: Vision and challenges," *IEEE Internet Things J.*, vol. 3, no. 5, pp. 637–646, Oct. 2016.

[3] R. Roman, J. Lopez, and M. Mambo, "Mobile edge computing, fog et al.: A survey and analysis of security threats and challenges," *Future Gener. Comput. Syst.*, vol. 78, pp. 680–698, Jan. 2016.

[4] J. Rivera and R. van der Meulen. (Mar. 2014). *Gartner Says the Internet of Things Will Transform the Data Center*. Accessed: Jun. 12, 2018. [Online]. Available https://iot.do/gartner-says-internet-things-will-transform-data-center-2014-03

[5] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures Commun. Netw.*, vol. 3, no. 1, pp. 1–211, 2010.

[6] A. Abdelnasser, E. Hossain, and D. I. Kim, "Clustering and resource allocation for dense femtocells in a two-tier cellular OFDMA network," *IEEE Trans. Wireless Commun.*, vol. 13, no. 3, pp. 1628–1641, Mar. 2014.

[7] S. Guruacharya, D. Niyato, M. Bennis, and D. I. Kim, "Dynamic coalition formation for network MIMO in small cell networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 10, pp. 5360–5372, Oct. 2013.

[8] S. Sardellitti, G. Scutari, and S. Barbarossa, "Joint optimization of radio and computational resources for multicell mobile-edge computing," *IEEE Trans. Signal Inf. Process. Over Netw.*, vol. 1, no. 2, pp. 89–103, Jun. 2015.

[9] J. Oueis, E. C. Strinati, S. Sardellitti, and S. Barbarossa, "Small cell clustering for efficient distributed fog computing: A multi-user case," in *Proc. IEEE 82nd Veh. Technol. Conf. (VTC-Fall)*, Sep. 2015, pp. 1–5.

[10] J. Oueis, E. C. Strinati, and S. Barbarossa, "The fog balancing: Load distribution for small cell cloud computing," in *Proc. Veh. Technol. Conf.*, May 2015, pp. 1–6.

[11] M. A. Islam, S. Ren, G. Quan, M. Z. Shakir, and A. V. Vasilakos, "Water-constrained geographic load balancing in data centers," *IEEE Trans. Cloud Comput.*, vol. 5, no. 2, pp. 208–220, Apr./Jun. 2015.

[12] Z. Liu, M. Lin, A. Wierman, S. Low, and L. L. H. Andrew, "Greening geographical load balancing," *IEEE/ACM Trans. Netw.*, vol. 23, no. 2, pp. 657–671, Apr. 2015.

[13] Q. Zhang, Q. Zhu, M. F. Zhani, R. Boutaba, and J. L. Hellerste, "Dynamic service placement in geographically distributed clouds," *IEEE J. Sel. Areas Commun.*, vol. 31, no. 12, pp. 762–772, Dec. 2013.

[14] X. Chen, L. Jiao, W. Li, and X. Fu, "Efficient multi-user computation offloading for mobile-edge cloud computing," *IEEE/ACM Trans. Netw.*, vol. 24, no. 5, pp. 2795–2808, Oct. 2016.

[15] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 974–983, Apr. 2015.

[16] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.

[17] M. Satyanarayanan, "Mobile computing: The next decade," in *Proc. 1st ACM Workshop Mobile Cloud Comput. Services, Soc. Netw. Beyond*, 2010, Art. no. 5.

[18] F. Bonomi, R. Milito, J. Zhu, and S. Addepalli, "Fog computing and its role in the Internet of Things," in *Proc. 1st Ed. MCC Workshop Mobile Cloud Comput.*, 2012, pp. 13–16.

[19] M. Satyanarayanan, P. Bahl, R. Caceres, and N. Davies, "The case for VM-based cloudlets in mobile computing," *IEEE Pervasive Comput.*, vol. 8, no. 4, pp. 14–23, Oct./Dec. 2009.

[20] A. Greenberg, J. Hamilton, D. A. Maltz, and P. Patel, "The cost of a cloud: Research problems in data center networks," *ACM SIGCOMM Comput. Commun. Rev.*, vol. 39, no. 1, pp. 68–73, 2009.

[21] TROPIC. (2015). *Distributed Computing Storage and Radio Resource Allocation Over Cooperative Femtocells*. [Online]. Available: http://www.ict-tropic.eu

[22] J. Rubio, A. Pascual-Iserte, J. del Olmo, and J. Vidal, "User association for load balancing in heterogeneous networks powered with energy harvesting sources," in *Proc. IEEE Globecom Workshops (GC Wkshps)*, Dec. 2014, pp. 1248–1253.

[23] H. H. M. Tam, H. D. Tuan, D. T. Ngo, T. Q. Duong, and H. V. Poor, "Joint load balancing and interference management for small-cell heterogeneous networks with limited backhaul capacity," *IEEE Trans. Wireless Commun.*, vol. 16, no. 2, pp. 872–884, Feb. 2017.

[24] M. Lin, Z. Liu, A. Wierman, and L. L. H. Andrew, "Online algorithms for geographical load balancing," in *Proc. Int. Green Comput. Conf. (IGCC)*, Jun. 2012, pp. 1–10.

[25] H. Xu, C. Feng, and B. Li, "Temperature aware workload managementin geo-distributed data centers," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 6, pp. 1743–1753, Jun. 2015.

[26] J. Luo, L. Rao, and X. Liu, "Spatio-temporal load balancing for energy cost optimization in distributed Internet data centers," *IEEE Trans. Cloud Comput.*, vol. 3, no. 3, pp. 387–397, Jul./Sep. 2015.

[27] Z. Liu, M. Lin, A. Wierman, S. H. Low, and L. L. H. Andrew, "Greening geographical load balancing," in *Proc. ACM SIGMETRICS Joint Int. Conf. Meas. Modeling Comput. Syst.*, 2011, pp. 233–244.

[28] F. Farahnakian, P. Liljeberg, and J. Plosila, "Energy-efficient virtual machines consolidation in cloud data centers using reinforcement learning," in *Proc. 22nd Euromicro Int. Conf. Parallel, Distrib. Netw.-Based Process. (PDP)*, Feb. 2014, pp. 500–507.

[29] X. Zhou, K. Wang, W. Jia, and M. Guo, "Reinforcement learning-based adaptive resource management of differentiated services in geo-distributed data centers," in *Proc. IEEE/ACM 25th Int. Symp. Qual. Service (IWQoS)*, Jun. 2017, pp. 1–6.

[30] N. Nisan, T. Roughgarden, E. Tardos, and V. V. Vazirani, *Algorithmic Game Theory*, vol. 1. Cambridge, U.K.: Cambridge Univ. Press, 2007.

[31] D. Kinderlehrer and G. Stampacchia, *An Introduction to Variational Inequalities and Their Applications*. Philadelphia, PA, USA: SIAM, 2000.

[32] R. B. Cooper, *Introduction to Queueing Theory*. Amsterdam, The Netherlands: North Holland, 1981.

[33] C. H. Liu, P. Hui, J. W. Branch, C. Bisdikian, and B. Yang, "Efficient network management for context-aware participatory sensing," in *Proc. 8th Annu. IEEE Conf. Sensor, Mesh Ad Hoc Commun. Netw.*, Jun. 2011, pp. 116–124.

[34] M. Mihailescu and Y. M. Teo, "On economic and computational-efficient resource pricing in large distributed systems," in *Proc. 10th IEEE/ACM Int. Conf. Cluster, Cloud Grid Comput.*, May 2010, pp. 838–843.

[35] Y. A. Korilis, A. A. Lazar, and A. Orda, "Capacity allocation under noncooperative routing," *IEEE Trans. Autom. Control*, vol. 42, no. 3, pp. 309–325, Mar. 1997.

[36] S. Katipamula *et al.*, "Small- and medium-sized commercial building monitoring and controls needs: A scoping study," Pacific Northwest Nat. Lab., Richland, WA, USA, Tech. Rep. PNNL-22169, Oct. 2012.

[37] Institute of Real Estate Management. (2012). *Trends in Office Buildings Operations*. Accessed: Jun. 12, 2018. [Online]. Available https://www.irem.org/File%20Library/IREM%20Store/Document%20Library/IESamples/12Samples/2012OfficeBuildTrends.pdf

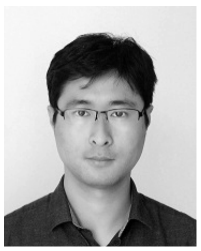
[38] Gridium. (2015) *Density Trends in Commercial Real Estate*. [Online]. Available: https://gridium.com/imagine-your-building-with-35-more-people-in-it/

[39] S. Penmatsa and A. T. Chronopoulos, "Game-theoretic static load balancing for distributed systems," *J. Parallel Distrib. Comput.*, vol. 71, no. 4, pp. 537–555, 2011.

[40] T. Pradeau, "Congestion games with player-specific cost functions," Ph.D. dissertation, Dept. Gen. Math., Univ. Paris-Est, Champs-sur-Marne, France, 2014.

Lixing Chen received the B.S. and M.S. degrees from the College of Information and Control Engineering, China University of Petroleum, Qingdao, China, in 2013 and 2016, respectively. He is currently pursuing the Ph.D. degree with the College of Engineering, University of Miami. His primary research interests include mobile edge computing, game theory, and machine learning for networks.

Sheng Zhou (S'06–M'12) received the B.E. and Ph.D. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2005 and 2011, respectively. In 2010, he was a Visiting Student with the Wireless System Laboratory, Department of Electrical Engineering, Stanford University, Stanford, CA, USA. From 2014 to 2015, he was a Visiting Researcher with the Central Research Laboratory, Hitachi, Ltd., Japan. He is currently an Associate Professor with the Department of Electronic Engineering, Tsinghua University. His research interests include mobile edge computing, vehicular networks, and green wireless communications.



Jie Xu (S'09–M'15) received the B.S. and M.S. degrees in electronic engineering from Tsinghua University, Beijing, China, in 2008 and 2010, respectively, and the Ph.D. degree in electrical engineering from UCLA in 2015. He is currently an Assistant Professor with the Electrical and Computer Engineering Department, University of Miami. His primary research interests include mobile edge computing, machine learning for networks, and network security.