Energy-Delay Tradeoff Analysis in Fog-Enabled D2D Network

Shuang Zhao[†,‡,*], Yang Yang[†,‡,*], Xiumei Yang[†,‡,*], Wuxiong Zhang[†,‡,*], Xiliang Luo[*,§], Hua Qian[¶]
[†]Shanghai Institute of Microsystem and Information Technology, CAS,
[‡]University of Chinese Academy of Sciences,
[*]Shanghai Institute of Fog Computing Technology (SHIFT),
[§]ShanghaiTech University,
[¶]Shanghai Advanced Research Institute, CAS

***Abstract*—Fog computing has recently emerged as a promising solution to liberate mobile devices from increasingly intensive computation workload, by offloading the computation task to the end-users or near user facilities. In order to achieve efficient computation offloading, it is crucial to take advantage of the network heterogenous communication and computation resources. To this end, in this paper, we propose a collaborative task offloading architecture based on a general fog-enabled device-to-device (D2D) network, where the computation and communication resources are dynamically shared among the mobile users. We focus on the average network power consumption minimization problem, meanwhile taking into account the signaling overhead and users' incentive to avoid the over-aggressive behaviors that harm the users' motivation for collaboration. Further, based on Lyapunov techniques, we develop an online D2D task offloading and resource scheduling algorithm, which dynamically determines the task offloading, CPU frequency scheduling and transmit power allocation for each user. Rigorous performance analysis is conducted, which characterizes a $[O(1/V), O(V)]$ tradeoff between the power consumption and the execution delay under the proposed algorithm in dynamic networks. Extensive simulation results demonstrate that the proposed online task offloading algorithm can offer better performance, in terms of the network power consumption and execution delay, than the user local execution scheme without offloading.**

I. Introduction

With the increasing popularity of smart hand-held devices, more and more mobile applications such as online gaming, 3D modeling and augmented reality are emerging and attracting great attention [2], [3]. This kind of applications are typically computation intensive, demanding higher level of data communication, computation and storage, which cannot be easily satisfied by general resource constrained mobile devices, e.g., the limited computation resources and battery capacity. To relax the the computation and energy consumption tension of mobile devices, computation offloading is a promising technology and emerging as a key focus in both academia and industry [4]–[6].

The basic idea of computation offloading is to shift the computation execution from the local devices to the resource-rich cloud infrastructure or nearby fog nodes. Computation offloading for mobile cloud computing has been widely studied in recent years, in which mobile users offload the computation to the remote network provider via wireless access [7]–[9]. Though the mobile cloud computing enriches the computation resources for mobile devices, it leads to serious transmission latency between the mobile devices and the clouds, which is always intolerable for the delay sensitive applications. As a good complement to mobile cloud computing, fog computing (also known as the extension of mobile edge computing (MEC)), offers computational capacity along the cloud to edge continuum [10]–[12], such as end-users or near-user facilities. Therefore, by offloading the computation tasks to the nearby fog nodes, mobile devices can enjoy low-latency as well as flexible computation experience.

In order to prolong the battery life and improve the computation performance, various of computation offloading policies have been proposed. Nevertheless, the efficient computation offloading depends on effective wireless data transmission, which highly depends on the wireless channel conditions. Therefore, joint optimization of communication and computation resource is essential for computation offloading policy design. In [13], an online joint radio and computational resource management algorithm for multi-user MEC system is proposed, in which the users offload the computation tasks to the proximate MEC server via cellular access. However, this additional traffic will aggravate the burden of the future cellular networks. In [14], the authors proposed a D2D task offloading framework where mobile users can leverage the computation resources of the nearby users according to their available computation capacity. It typically assumed that the collaborated users in the network trust each other, which is impractical in practice. In [15], a novel energy efficient D2D Fogging framework is proposed, which jointly considered user incentive and collaboration into task offloading optimization. However, the downside is that it focused on network wide energy minimization, while not taking the execution delay into account, which is a key issue in fog computing.

In this paper, we consider a fog-enabled D2D network, where mobile users can dynamically and beneficially share the computation and communication resources among each other via the assistance of the centralized network controller. We focus on designing an online energy-efficient D2D task offloading and resource scheduling policy, while taking the execution delay into account. In each time slot, the system operations including dynamic task offloading policy, CPU frequencies

Part of this work will be presented at IEEE Fog World Congress (FWC), Santa Clara, CA, Oct. 2017 [1].

scheduling and transmit power allocation are determined by the centralized network controller with global information according to the current network states and users' task arrivals. Then, each mobile users will execute the computation tasks locally or offload part of them via a D2D link to a nearby user.

There are several challenges need to be addressed in designing the task offloading and resource scheduling policy. First, a stochastic system is considered where user locations, available D2D connections, traffic demands, and channel conditions change over the time. This requires the centralized network controller to design an online algorithm, which determines D2D task offloading and resource scheduling policy dynamically. Besides, the frequent D2D communication among the users certainly will cause high signaling overhead and reduce the offloading efficiency. Therefore, there exists a tradeoff between the signaling overhead reduction and the offloading capacities. In addition, to prevent the over-aggressive behaviors that harm user's motivation for collaboration, a reasonable incentive scheme also should be provided. Moreover, the centralized network controller needs to reduce the network power consumption while maintaining the delay guarantee for each user, which requires the centralized network controller to make a good balance between the network power consumption and the delay of users.

To tackle these challenges, we propose the online computation efficient algorithm for D2D task offloading and resource scheduling in this paper. The main contributions of this paper are summarized as follows.

- Based on a general fog-enabled D2D network, we propose a collaborative task execution architecture, which simultaneously takes user's incentive and signaling overhead into account. Further, we formulate an average network power consumption minimization problem, along with users' incentive and task buffer stability constraints.
- Based on Lyapunov optimization techniques, we develop an online (real time) D2D task offloading and resource scheduling algorithm, which jointly optimize the network heterogeneous communication and computing capacities among the mobile users. Specifically, in each time slot, the network operation is determined by solving a deterministic problem, where the CPU frequencies are obtained in closed form and the D2D task offloading and transmit power allocation are obtained through the efficient Gauss-Seidel method.
- Performance analysis is conducted for the proposed algorithm, which explicitly characterizes the power-delay tradeoff. Compared with the user local execution, simulation results show that the proposed algorithm can offer much better performance, in terms of the average network power consumption and average delay.

The rest of the paper is organized as follow. We introduce the system model and formulate the average network power consumption minimization problem in Section II and Section III, respectively. We develop an online D2D task offloading and resource scheduling algorithm and provide the corresponding theoretical analysis in Section IV. Simulation results for the proposed algorithm are shown in Section V. Section VI concludes this paper.

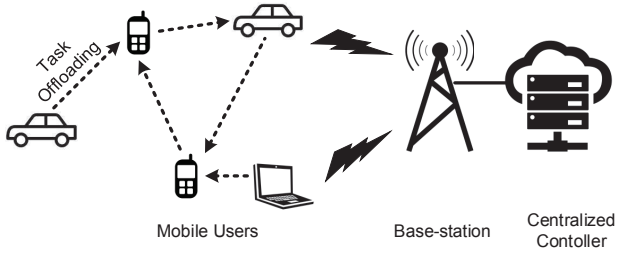


Fig. 1: Illustration of Fog-enabled D2D network.

II. System Model

We consider a fog-enabled D2D network as shown in Fig.1, where a set of $\mathcal{U} = \{1, \cdots, |\mathcal{U}|\}$ mobile users are running computation intensive tasks under the assistance of the centralized network controller. The centralized network controller could be a small data center installed at the base station deployed by the telecom operator, thus it can be accessed by the mobile users through wireless channels and realize the computation offloading management and resource scheduling with the readily available wireless channel state information at the base station. Each mobile user is implemented with a frequency tunable CPU and can be viewed as a mobile fog node in the network. The computation and communication resources are shared among those fog nodes, which means that through establishing D2D links with nearby users, each mobile user can opportunistically leverage the under-utilized computation resources to execute the computation tasks. By offloading part of the computation tasks to the nearby fog nodes with sufficient computation resources, the mobile users can not only enjoy a higher quality of experience (QoE), but also reduce the battery energy consumption [13].

For convenience, we assume that the network operates in a slotted structure, indexed by $t \in \{0, 1, 2, \cdots\}$ and the time slot length is τ. Motivated by the fact that the centralized network controller has the global network information, including users' location, computational resources and task arrivals, we consider a network-assisted architecture where in each time slot the centralized network controller conducts resource discovery, D2D connections, and task offloading scheduling for each mobile user. We should emphasize here that in our model, the wireless links between the mobile users and the base station, transmit the control information to the users only. To be clear, the computation offloading from the mobile users to the base station is not considered in this paper. Under this conditions, we can see more clearly about the benefit of the D2D task offloading and liberate the burden of the cellular network.

A. Local Execution Model

The local computation capacity for each mobile user depends on the local CPU frequency, which is scheduled by the centralized network controller in each time slot. Denote

the CPU frequency of user i in time slot t as $f_i(t)$, which is constrained by the maximum value $f_i^{\max}$. Define $\boldsymbol{f}(t) \triangleq \big[f_1(t), \cdots, f_{|\mathcal{U}|}(t)\big]$, $0 \le f_i(t) \le f_i^{\max}$.

Denote $\mu_i^l(t)$ as the amount of computation tasks that executed locally at the ith mobile user in time slot t, thus $\mu_i^l(t)$ can be expressed by

$$\mu_i^l(t) = \tau f_i(t) L_i^{-1}, \tag{1}$$

where L_i is the number of CPU cycles needed for processing one bit task at ith mobile users and can be obtained by off-line measurements [16].

Accordingly, the power consumption for local computation execution model at user i can be modeled as:

$$p_i^l(t) = \kappa f_i^3(t), \tag{2}$$

where κ is the effective switched capacitance related to the chip architechture [17].

B. D2D Offloading Model

In each time slot, the centralized network controller processes the device discovery for a user to detect the set of its nearby connectable fog nodes, and then establish the association between users according to the current network states. Denote the distance between user i and j in time slot t as $d_i^j(t)$ and denote the maximum connectable D2D communication distance as $d_i^{\max}$, $i \in \mathcal{U}$.

Note that the feasible D2D links among the users varying across different time slot due to users' mobility, we introduce a time-varying D2D connectivity graph $\mathcal{G}(t) = \{\mathcal{U}, \mathcal{E}(t)\}$, where $\mathcal{E}(t) = \{(i,j) | i \ne j, i, j \in \mathcal{U}, d_i^j(t) \le d_i^{\max}\}$ contains all feasible D2D links in time slot t.

Denote $x_{ij}(t)$ as the binary task offloading indicator between mobile user i and j. Specially, $x_{ij}(t) = 1$ if a user i offloads its computation task to execute on user j in time slot t, and 0 otherwise. In addition, $x_{ij}(t)$ is asymmetric, i.e., $x_{ij}(t) \ne x_{ji}(t)$. Define $\boldsymbol{x}(t) \triangleq \{x_{ij}(t) | \forall i, j \in \mathcal{U}\}$.

We assume that the mobile users in the network establish the D2D links with each other through *space-devision multiple access* (SDMA) and each mobile user is implemented with M antennas, which implies that each user can receive the offloaded computation task from at most M fog nodes simultaneously. We further assume that each user can offload part of its computation task to at most one another fog node in a time slot. Thus $x_{ij}(t)$ should be chosen from the following feasible set $\mathcal{A}$,

$$\mathcal{A} \triangleq \left\{ x_{ij} \in \{0,1\} \middle| \begin{array}{l} x_{ij} = 0, \ \forall \ (i,j) \notin \mathcal{E}(t); \\ \sum_{j \in \mathcal{U}} x_{ij} \le 1, \ \forall i \in \mathcal{U}; \\ \sum_{i \in \mathcal{U}} x_{ij} \le M, \ \forall j \in \mathcal{U}. \end{array} \right\}.$$

The wireless channels between mobile users are assumed to be flat fading [13]. Denote the small scale fading gain between user i and j as $s_i^j(t)$, which follows Rayleigh distribution and can be viewed as constant in time slot t. The channel power gain between them is given by $G_i^j(t) = s_i^j(t) g_0 (d_0 \backslash d_i^j(t))^\theta$, where g_0 is pass-loss constant, d_0 is the reference distance and θ is the pass-loss exponent.

Note that in practice, two mobile users needs to negotiate with each other before executing task offloading between them. Thus the signaling overhead exists before executing the task offloading [18]. We introduce α as the *signaling overhead ratio*, which represents the occupied percentage of computation resource for establish D2D communication links. Therefore, the amount of computation tasks that can be offloaded from user i to user j in time slot t is given by

$$\mu_i^j(t) = (1-\alpha)\omega_i \tau \log_2 \left(1 + \frac{p_i^d(t) G_i^j(t)}{\omega_i N_0}\right), \tag{3}$$

where $0 \le \alpha \le 1$, $p_i^d(t)$ is the transmit power allocated by the user i for D2D transmission with maximum value $p_i^{d,\max}$, ω_i is the available bandwidth at the user i for D2D transmission. Define $\boldsymbol{p}^d(t) \triangleq [p_1^d(t), \cdots, p_N^d(t)]$ as the power allocation vector for D2D transmission.

Therefore, the amount of computation task that offloaded by user i through D2D link in time slot t can be expressed by

$$\mu_i^d(t) = \sum_{j \in \mathcal{U}} \mu_i^j(t) x_{ij}(t). \tag{4}$$

C. Incentive Constraint Model

For each user, the computation tasks waiting to be executed locally include two parts, the locally generated computation tasks and the offloaded computation tasks from other users. Specifically, denote $\mu_i^{l,j}(t)$ as the amount of local computation resource of user i scheduled for user j's task execution in time slot t, according to a certain scheduling discipline, such as FIFO, LIFO or Random discipline. Note that $\mu_i^{l,i}(t)$ indicates the amount of computation resource allocated for computation tasks generated locally. We adopt fully-efficient scheduling policy, which means:

$$\sum_{j \in \mathcal{U}} \mu_i^{l,j} = \mu_i^l(t), \ \forall i \in \mathcal{U}. \tag{5}$$

Next, we denote β as the proportion of local resource scheduled for other users. The parameter β is within $[0,1]$ and is called incentive factor hereafter. Therefore, we have

$$\sum_{j \in \mathcal{U}, j \ne i} \mu_i^{l,j} = \beta \mu_i^l(t), \ \forall i \in \mathcal{U}. \tag{6}$$

To encourage mobile users to participate in D2D offloading in long run and prevent the users from over-aggressive manners, we introduce the incentive constraint which ensures that a mobile user exploits more resources from other fog nodes only if it contributes more resources for others [15]. The incentive constraint is defined as follow:

$$\overline{\mu}_i^d(t) \le \beta \overline{\mu}_i^l(t), \ \forall i \in \mathcal{U}, \tag{7}$$

where $\overline{\mu}_i^d(t)$ and $\overline{\mu}_i^l(t)$ stand for time averages of $\mu_i^d(t)$ and $\mu_i^l(t)$ for each $i \in \mathcal{U}$. The incentive constraint reflects the amount of resource that mobile users exploit from the others is in proportion to that they contribute to the others. If they want more, they need to share more in return.

D. User Traffic and Queueing Model

In each time slot, $A_i(t)$ (bits) of computation tasks arrive at the ith user, which consists of two parts. One is the task offloaded from other users through D2D links in the previous slot, denoted as $a_i^d(t)$. The other one is the computation tasks generated locally, denoted as $a_i^l(t)$. $a_i^l(t)$ is assumed to be *i.i.d.* with $\mathbb{E}\{a_i^l(t)\} = \lambda_i$, $i \in \mathcal{U}$ and $0 \leq a_i^l(t) \leq a_i^{\max}$. Thus the total amount of arrived computation tasks $A_i(t)$ is given by $A_i(t) = a_i^l(t) + a_i^d(t)$, which can be processed starting from the $(t+1)$th time slot.

In each time slot, $\mu_i^l(t)$ bits of computation tasks will be executed at local CPU, while μ_i^d bits of computation tasks will be offloaded to the other users. The arrived but not executed tasks will be queued in the task buffers at each mobile device. We assume that each mobile user $i \in \mathcal{U}$ has $|\mathcal{U}|$ task buffers $\{Q_{ij}(t), j \in \mathcal{U}\}$. Specifically, $Q_{ii}(t)$ denotes the queue length of task buffer at user i cached the unexecuted computation tasks generated locally, which evolves according to the following equation:

$$Q_{ii}(t+1) = \max\Big[Q_{ii}(t) - (1-\beta)\mu_i^l(t) - \mu_i^d(t), 0\Big] + a_i^l(t). \tag{8}$$

$Q_{ij}(t)$ denotes queue length of task buffer cached the unexecuted tasks offloaded by user j in previous slots, which updates as follows:

$$Q_{ij}(t+1) = \max\Big[Q_{ij}(t) - \mu_i^{l,j}(t), 0\Big] + a_i^d(t), \forall j \in \mathcal{U}\backslash i, \tag{9}$$

where $a_i^d(t)$ is the newly arrived computation task from other users in time slot t and is given by

$$a_i^d(t) = \sum_{j\in\mathcal{U}} \max\Bigg\{\min\Big\{Q_{jj}(t) - (1-\beta)\mu_j^l(t), \mu_j^d(t)\Big\}, 0\Bigg\} x_{ji}(t). \tag{10}$$

Denote the sum queue length of the total task buffers at beginning of the tth time slot as $Q_i^{\text{sum}}(t)$, $Q_i^{\text{sum}}(t) = \sum_{j\in\mathcal{U}} Q_{ij}(t)$. Define $\boldsymbol{Q}^T(t) \triangleq [Q_1^{\text{sum}}(t), \cdots, Q_{|\mathcal{U}|}^{\text{sum}}(t)]$. By summing over (8) and (9) for each user $i \in \mathcal{U}$, we have

$$Q_i^{\text{sum}}(t+1) = \max\big[Q_i^{\text{sum}}(t) - C_i(t), 0\big] + A_i(t), \tag{11}$$

where $C_i(t) = \mu_i^l(t) + \mu_i^d(t)$ with the maximum value $C_{\max}$ and $A_i(t) = a_i^l(t) + a_i^d(t)$ with the maximum value $A_{\max}$.

The queueing process $Q_i^{\text{sum}}(t)$ is stable when the following condition holds [19],

$$\overline{Q}_i^{\text{sum}} = \lim_{t\to\infty} \frac{1}{t} \sum_{k=0}^{t-1} \mathbb{E}\{Q_i^{\text{sum}}(k)\} < \infty. \tag{12}$$

III. Problem Formulation

In this section, we first introduce the performance metrics, which are the time-averaged network wide power consumption and the time-averaged delay per user. Then an averaged network power consumption minimization problem with task buffer queues stability and incentive constraints will be formulated.

A. Performance Metrics

Since the energy issue is a key concern and the bottleneck for mobile users [15], the main objective in this paper is to make the best effort to reduce the total power consumption of all the users in the network. We focus on the power consumption for task execution as well as the transmit power for task offloading. Energy consumed at the centralized network controller is ignored for simplicity. The time-averaged power consumption of different mobile users in the network is adopted as the performance metric, which is defined as follows:

$$P_{av} \triangleq \overline{P} = \lim_{t\to\infty} \frac{1}{t} \mathbb{E}\Big[\sum_{k=0}^{t-1} \sum_{i\in\mathcal{U}} P_i(k)\Big], \tag{13}$$

where $P_i(t) \triangleq p_i^l(t) + p_i^d(t)$.

According to the Little's Law [20], the average traffic delay experienced by each mobile user is proportional to the averaged unexecuted computation tasks, which is the average sum queue length of task buffers. Thus metric of the average delay per user can be computed as follows,

$$D_{av} \triangleq \frac{\sum_{i\in\mathcal{U}} \overline{Q}_i^{\text{sum}}}{\sum_{i\in\mathcal{U}} \lambda_i} \tag{14}$$

B. Average Power Consumption Minimization

Denote the centralized network controller operation in time slot t as $\mathcal{O}(t) \triangleq \{\boldsymbol{f}(t), \boldsymbol{x}(t), \boldsymbol{p}^d(t)\}$. Thus the averaged network power consumption minimization problem is formulated below,

$$\begin{aligned}
\mathcal{P}1: \min_{\mathcal{O}(t)} \quad & \overline{P} && (15)\\
\text{s.t.} \quad & \overline{\mu}_i^d(t) \leq \beta \overline{\mu}_i^l(t), \ \forall i \in \mathcal{U} && (16)\\
& \overline{Q}_i^{\text{sum}} < \infty, \ \forall i \in \mathcal{U} && (17)\\
& 0 \leq f_i(t) \leq f_i^{\max}, \ u \in \mathcal{U} && (18)\\
& 0 \leq p_i^d(t) \leq p_u^{d,\max}, \ i \in \mathcal{U} && (19)\\
& x_{ij}(t) \in \mathcal{A}, \ \forall i,j \in \mathcal{U}, && (20)
\end{aligned}$$

where (18) is the scheduled CPU frequency constraint, and (19) is the allocated transmit power constraint.

Remark 1: Note that $\mathcal{P}1$ is a stochastic optimization problem, for which, the CPU frequency scheduling for each mobile users, D2D task offloading policy, and the transmit power allocation need to be determined in each time slot. This is a highly challenging problem with a large amount of stochastic information to be handled. Besides, the optimal decisions are temporally correlated due to the randomly traffic arrivals.

In the following, we focus on solving $\mathcal{P}1$ by using the Lyapunov optimization techniques [19], in which a deterministic per-slot problem is solved within each time slot. In addition, we show that the proposed online D2D task offloading and resource scheduling algorithm guarantees a low bound $1/2$ of the optimal value and reveal the network power consumption versus delay tradeoff.

IV. Online Task Offloading and Resource Scheduling Policy

In this section, we propose an online Task Offloading and Resource Scheuling Policy to solve $\mathcal{P}1$ by invoking the Lyapunov optimization technique [19], and then provide the performance analysis for the proposed algorithm.

A. Lyapunov Optimization Based Algorithm

The key idea of Lyapunov optimization is to turn time-average constraints into a pure queue stability problem. Following this idea, we first introduce virtual queue $Z_i(t)$ for each $i \in \mathcal{U}$, which is used to enforce the incentive constraint $\overline{\mu}_i^d(t) \leq \beta \overline{\mu}_i^l(t)$ and evolves as follow:

$$Z_i(t+1) = \max\Big[Z_i(t) + \mu_i^d(t) - \beta\mu_i^l(t), 0\Big]. \tag{21}$$

Define $\boldsymbol{Z}^T(t) \triangleq [Z_1(t), \cdots, Z_{|\mathcal{U}|}(t)]$ be the virtue queue vector. Next we define the quadratic Lyapunov function as:

$$L(\boldsymbol{\Theta}(t)) \triangleq \frac{1}{2}\sum_{i\in\mathcal{U}} Q_i^{\text{sum}}(t)^2 + \frac{1}{2}\sum_{i\in\mathcal{U}} Z_i(t)^2, \tag{22}$$

where $\boldsymbol{\Theta}(t)$ is a concatenated vector for all actual and virtual queues, $\boldsymbol{\Theta}(t) = [\boldsymbol{Q}^T(t), \boldsymbol{Z}^T(t)]$.

We further define the following conditional Lyapunov drift function $\Delta(\boldsymbol{\Theta}(t))$ as below,

$$\Delta(\boldsymbol{\Theta}(t)) \triangleq \mathbb{E}\Big\{L(\boldsymbol{\Theta}(t+1)) - L(\boldsymbol{\Theta}(t))|\boldsymbol{\Theta}(t)\Big\}, \tag{23}$$

which is the change of the Lyapunov function from one time slot to the next. The expression $\mathbb{E}\{\text{x}\}$ means the expectation of x. Accordingly, the one-slot conditional Lyapunov drift-plus-penalty function is shown as below:

$$\Delta_V(\boldsymbol{\Theta}(t)) = \Delta(\boldsymbol{\Theta}(t)) + V\sum_{i\in\mathcal{U}}\mathbb{E}\{P_i(t)|\boldsymbol{\Theta}(t)\}, \tag{24}$$

where V is a non-negative control parameter that balances the network power consumption and queue backlogs.

The following lemma provides an upper bound of the drift-plus-penalty function, which plays a significant role in solving the problem $\mathcal{P}1$.

Lemma 1*:* For arbitrary control policy $\mathcal{O}(t)$ for all time slots t, the drift-plus-penalty function $\Delta_V(\boldsymbol{\Theta}(t))$ satisfies:

$$\begin{aligned}\Delta_V(\boldsymbol{\Theta}(t)) \;\leq\; & \mathcal{K} + V\sum_{i\in\mathcal{U}}\mathbb{E}\big\{P_i(t)|\boldsymbol{\Theta}(t)\big\} \\ & + \sum_{i\in\mathcal{U}}\mathbb{E}\big\{Q_i^{\text{sum}}(t)[A_i(t) - C_i(t)]|\boldsymbol{\Theta}(t)\big\} \\ & + \sum_{i\in\mathcal{U}}\mathbb{E}\big\{Z_i(t)[u_i^d(t) - \beta\mu_i^l(t)]|\boldsymbol{\Theta}(t)\big\}, \end{aligned}\tag{25}$$

where $\mathcal{K}$ is a constant .

proof: Please refer to Appendix A. ■

The main principle of the Lyapunov optimization techniques [19] is to minimize the upper bound of Lyapunov drift-plus-penalty, i.e., the right-hand-side of (25). By doing so, all queue backlogs are consistently pushed towards a small size, meanwhile, the network power consumption can be minimized. The proposed Online Task Offloading and Resource Scheduling algorithm is summarized in Algorithm 1, in which a deterministic optimization problem $\mathcal{P}2$ needs to be solved in each time slot. Note that the optimization problem $\mathcal{P}2$ corresponds to the upper bound of $\Delta_V(\boldsymbol{\Theta}(t))$ minimization, where the terms that are not affected by $\mathcal{O}(t)$ are omitted in the objective function of $\mathcal{P}2$. The solution of $\mathcal{P}2$ will be presented in next subsection.

Algorithm 1 The Online D2D Task Offloading and Resource Scheduling Algorithm

1: Set t = 0, $\boldsymbol{\Theta}(0) = 0$;
2: **While** t $<$ t_{end}, **do**
3: At beginning of the tth time slot, observe random events $\{a_i^l(t)\}$, $\{G_i^j(t)\}$ and feasible D2D links $\mathcal{E}(t)$. Then compute $\{Q_i^{\text{sum}}(t)\}$ and $\{Z_i(t)\}$;
4: Determine $\boldsymbol{f}(t), \boldsymbol{x}(t), \boldsymbol{p}^d(t)$ by solving

$$\begin{aligned}\mathcal{P}2: \min_{\mathcal{O}(t)} \quad & V\sum_{i\in\mathcal{U}} P_i(t) + \sum_{i\in\mathcal{U}} Z_i(t)[\mu_i^d(t) - \beta\mu_i^l(t)] \\ & + \sum_{i\in\mathcal{U}} Q_i^{\text{sum}}(t)[a_i^d(t) - \mu_i^l(t) - \mu_i^d(t)] \\ \text{s.t.} \quad & x_{ij}(t) \in \mathcal{A}, \ (18), \text{ and } (19)\end{aligned}$$

5: Update $Q_{ij}(t+1)$ according to (8) and (9);
6: t $\leftarrow$ t + 1.
7: **end While**

B. Solutions for $\mathcal{P}2$

In this subsection, we will develop the online CPU frequency scheduling, task offloading and transmit powers allocation for $\mathcal{P}2$ which can be obtained by solving two subproblems.

1) Optimal CPU frequency scheduling: According to the definitions of $P_i(t)$, $p_i^l(t)$, and $\mu_i^l(t)$, we rearrange the objective function in $\mathcal{P}2$ and decouple $f_i(t)$ from $\mathcal{P}2$. Then the optimal CPU frequency scheduling for user $i \in \mathcal{U}$ in time slot t can be obtained by solving following subproblem $\mathcal{SP}_1$:

$$\begin{aligned}\mathcal{SP}_1: \min_{f_i(t)} \quad & \sum_{i\in\mathcal{U}}\Big[V\kappa f_i^3(t) - (Q_i^{\text{sum}}(t) + \beta Z_i(t))\tau f_i(t) L_i^{-1}\Big] \\ \text{s.t.} \quad & 0 \leq f_i(t) \leq f_i^{\max}, \ i \in \mathcal{U},\end{aligned}$$

which is a convex optimization problem and the optimal CPU frequency scheduling for user i is given by

$$f_i^\star(t) = \min\Big\{f_i^{\max}, \sqrt{\frac{(Q_i\text{sum}(t) + \beta Z_i(t))\tau}{3\kappa V L_i}}\Big\}, \ i \in \mathcal{U}. \tag{26}$$

Remark 2*:* Note that the local CPU frequency scheduling $f_i^\star(t)$ is controlled by $Q_i^{\text{sum}}(t)$, $Z_i(t)$, V and L_i simultaneously. $f_i^\star(t)$ increases with the increase of $Q_i^{\text{sum}}(t)$ and $Z_i(t)$. The increase of $Q_i^{\text{sum}}(t)$ or $Z_i(t)$ implies that the user needs to execute more computation task to keep the task buffers stable, thus it requires larger CPU frequency. Besides, $f_i^\star(t)$ decreases with the increase of V and L_i. Larger V means the larger weight of power consumption term in the objective function of $\mathcal{SP}_1$, which leads to a lower local CPU frequency allocation to reduce the power consumption. Similarly, larger

$$\mathcal{SP}_2: \min_{\boldsymbol{x}(t),\boldsymbol{p}^d(t)} \quad V\sum_{i\in\mathcal{U}} p_i^d(t) + \sum_{i\in\mathcal{U}} Q_i^{\text{sum}}(t)\left[\sum_{j\in\mathcal{U}}\mu_j^i(t)x_{ji}(t) - \sum_{j\in\mathcal{U}}\mu_i^j(t)x_{ij}(t)\right] + \sum_{i\in\mathcal{U}} Z_i(t)\sum_{j\in\mathcal{U}}\mu_i^j(t)x_{ij}(t) \tag{28}$$
$$\text{s.t.} \quad x_{ij}(t)\in\mathcal{A},\ \forall i,j\in\mathcal{U}$$
$$0\le p_i^d(t)\le p_i^{d,\max},\ \forall i\in\mathcal{U}.$$

$$\mathcal{P}_{\text{pw}}: \min_{p_i^d(t)} \quad Vp_i^d(t) - \left[Q_i^{\text{sum}}(t) - Z_i(t) - \sum_{j\in\mathcal{U}} Q_j^{\text{sum}}(t)x_{ij}(t)\right]\sum_{j\in\mathcal{U}}\mu_i^j(t)x_{ij}(t) \tag{29}$$
$$\text{s.t.} \quad 0\le p_i^d(t)\le p_i^{d,\max}.$$

$$p_i^{d,\star}(t) = \min\left\{p_i^{d,\max}, \max\left\{\frac{\left[Q_i^{\text{sum}}(t) - Z_i(t) - \sum_{j\in\mathcal{U}} Q_j^{\text{sum}}x_{ij}(t)\right](1-\alpha)\omega_u\tau}{V\ln 2} - \frac{N_0\omega_u}{\sum_{j\in\mathcal{U}} G_i^j(t)x_{ij}(t)}, 0\right\}\right\} \tag{30}$$

L_i means the lower efficiency of CPU frequency, thus leads to a lower local CPU frequency directly.

2) Optimal Task Offloading and Transmit Power allocation: By the definition of $a_i^d(t)$ in (10) , we have the following inequality

$$a_i^d(t) \le \sum_{j\in\mathcal{U}} \mu_j^i(t)x_{ji}(t), \tag{27}$$

After decoupling $\boldsymbol{f}(t)$ from $\mathcal{P}2$, we invoke the inequality above and $\mu_i^d(t) = \sum_{j\in\mathcal{U}}\mu_i^j(t)x_{ij}(t)$ into the objective function of $\mathcal{P}2$. Then the D2D task offloading indicator $\boldsymbol{x}(t)$ and transmit power allocation $\boldsymbol{p}^d(t)$ can be obtained by solving the subproblem $\mathcal{SP}_2$.

It is not difficult to identify that $\mathcal{SP}_2$ is a mixed integer programming problem involving two sets of optimization variables: task offloading indicator and transmit power allocation. The computation complexity is prohibitively high for brute force. By further exploiting the structure information of (28), we found that the feasible region is a Cartesian product of those of $\boldsymbol{p}^d(t)$ and $\boldsymbol{x}(t)$. *Gauss-seidel method* is an time efficient method for such an optimization problem [16], [21], [22], which minimizes the task offloading indicator and power allocation in an alternating manner and guarantees the convergence. In each iteration, the optimal transmit power allocation is obtained in closed form and the the task offloading indicator is determined by combinatorial optimization. Next, we derive the solution in each iteration of the ObjectiveSP22 for transmit power allocation and D2D task offloading.

Transmit Power Allocation : For a fixed task offloading indicator $\boldsymbol{x}(t)$, the optimal transmit power of mobile user i can be obtained by solving the problem $\mathcal{P}_{\text{pw}}$, which is a convex optimization problem and the optimal transmit power $p_i^{d,\star}(t), i\in\mathcal{U}$ is given by (30).

D2D Task Offloading: For a fixed transmit power allocation $\boldsymbol{p}^d(t)$, rearrange the terms involving $x_{ij}(t)$ in (28) and omit the terms that are not affected by $\boldsymbol{x}(t)$, the suboptimal task offloading can be obtained by solving the problem $\mathcal{P}_{\text{ua}}$. To simplify the notation, we introduce the definition $W_{ij}(t)$ as shown in (31), which is a constant during per-slot.

Through the following *Lemma 2*, we demonstrate that the problem $\mathcal{P}_{\text{ua}}$ can be formulated as a normalized modular maximization function with two partition matroid constraints, the structure of which can be exploited to design computationally efficient algorithms for the problem $\mathcal{P}_{\text{ua}}$ with provable approximation gaps.

Lemma 2: The problem $\mathcal{P}_{\text{ua}}$ can be formulated as the maximization of a normalized modular function subject to two partition matroid constraints on ground set $\mathcal{E}(t)$.

Proof: Please refer to Appendix B. ■

Normally, the greedy algorithm provides an effective solution for such a modular maximization problem [23]. Such algorithm guarantees a tight $\delta = \frac{1}{2}$-approximation gap for the problem $\mathcal{P}_{\text{ua}}$, i.e., the worst case is at least 50% of the optimal solution. The greedy algorithm is described in Algorithm 2. It stars with an empty task offloading set $\boldsymbol{x} = \emptyset$ and a initial set $\mathcal{E}' = \mathcal{E}(t)$. Then the greedy algorithm iteratively selects the element $(i^\star, j^\star)$ from the set $\mathcal{E}'$ with the highest marginal value $W_{i^\star j^\star}(t)$ while maintains the matroid constraints in each step. Once the element $(i^\star, j^\star)$ is added to $\boldsymbol{x}$, both $i^\star$ and $j^\star$ should be deleted from set $\mathcal{E}'$. The greedy algorithm stops until the largest marginal value $W_{i^\star j^\star}(t)$ is 0.

Algorithm 2 The Greedy Algorithm

1: Initialize $\boldsymbol{x} = \emptyset$ and $\mathcal{E}' = \mathcal{E}(t)$;
2: **Repeat** Select $(i^\star, j^\star) \in \mathcal{E}'$ as the solution to $\max_{\{\boldsymbol{x}\cup(i,j)\}\in\mathcal{A}} W_{ij}(t)$;
3: Update $\boldsymbol{x} = \{\boldsymbol{x}\cup(i^\star, j^\star)\}$;
4: Update $\mathcal{E}' = \mathcal{E}'\backslash(i^\star, j^\star)$;
5: **Until** $W_{i^\star j^\star}(t) = 0$;
6: **Output** $\boldsymbol{x}$.

C. Performance Analysis

In this subsection, we present the main theoretical result of this paper, which characterize the lower bound for the network average power consumption and the average delay per user

$$\mathcal{P}_{\text{ua}}: \max_{\boldsymbol{x}(t)\in\mathcal{A}} \sum_{i\in\mathcal{U}}\sum_{j\in\mathcal{U}} \underbrace{\left[Q_i^{\text{sum}}(t)-Q_j^{\text{sum}}(t)-Z_i(t)\right](1-\alpha)\omega_i\tau\log_2\left(1+\frac{p_i^d(t)G_i^j(t)}{\omega_i N_0}\right)}_{\triangleq W_{ij}(t)} x_{ij}(t) \tag{31}$$

and reveal the tradeoff between the network average power consumption and average delay.

***Theorem 1**:* For the network defined in section II, we have:

1) The average power consumption of mobile users under the proposed the dynamic online D2D task offloading and resource scheduling algorithm satisfies:

$$P_{av} \le P_\delta^{\text{opt}} + \frac{\mathcal{K}}{V}, \tag{32}$$

where P_δ^{opt} is the suboptimal value of $\mathcal{P}1$ with $\delta = \frac{1}{2}$.

2) Assume there exist constants $\epsilon > 0$ and $\Psi_\epsilon$which satisfy following Slater Type Conditions in [19], then the average delay experienced by each mobile user under the proposed algorithm satisfies:

$$D_{av} \le \frac{\mathcal{K} + V(\Psi_\epsilon - P^{\text{opt}})}{\delta\epsilon\sum_i \lambda_i}. \tag{33}$$

Proof: Please refer to Appendix C.

***Remark 2**:* Theorem 1 demonstrates that under the proposed online D2D task offloading and resource scheduling algorithm, the average network power consumption decreases inversely proportional to V, and then converges to the suboptimal value when V is sufficient large. Meanwhile, the average delay per user increases linearly with V. Hence, there exists an $[O(1/V), O(V)]$ tradeoff between those two objects, which provide us a valuable guideline in practical implementations. Specifically, for the delay-sensitive networks, we set V to a small value, while for the energy- sensitive and delay-tolerant networks, we can set V to a large value.

V. Numerical Result

In this section, we evaluate the performance of proposed online task offloading algorithm through numerical simulations. We consider a network with $|\mathcal{U}|$ randomly deployed mobile users. The cover area of the network is $150 \times 150\ m^2$ and the maximum D2D communication distance is $d_i^{\max} = 20\ m$ for all $i \in \mathcal{U}$. We assume that the small fading channel power gains are with unit mean. Besides, we set $g_0 = -40$ dB, $\theta = 5$, $d_0 = 1\ m$ and $N_0 = -174$ dB/Hz. We also set $w_i = 10$ MHz, $\kappa = 10^{-27}$, $L_i = 500$ cycles/bit, $p_i^{d,\max} = 200$ mW and $f_i^{\max} = 2$ GHz for all mobile users $i \in \mathcal{U}$. The simulation result are averaged over 3000 constant time slots with slot length $\tau = 1$ ms.

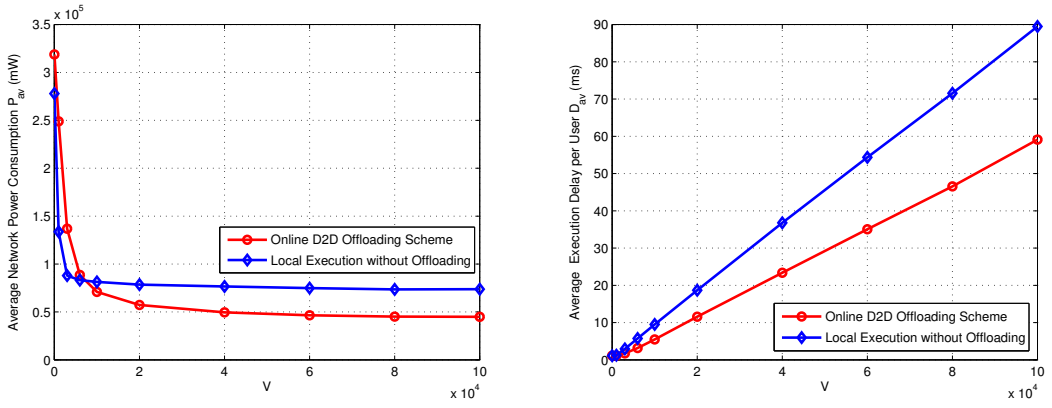
We first validate the theoretical results derived in Theorem 1 for the proposed online D2D task offloading and resource scheduling algorithm in Fig. 2. The signaling overhead ratio is set to $\alpha = 0.05$ and the incentive factor is set to $\beta = 0.2$. It can be observed from Fig. 2(a) that the average network power consumption decreases inversely proportion to the control parameter V and converges to a constant when V is sufficient large. Meanwhile, as shown in Fig. 2(b), the average delay per user experienced increases linearly with V. Thus these observations verify that there is an $[O(1/V), O(V)]$ tradeoff between average network power consumption and average delay, and indicates that proper V should be chosen to balance these two objects.

Fig. 2 also compares the online D2D offloading scheme in the proposed algorithm with the scheme where the each user executes its computation task locally without offloading. We observe from Fig. 2(a) that the online D2D offloading scheme outperforms the local execution scheme when $V > 6 \times 10^3$ in terms of the average network power consumption. Specifically, the additional average power reduction in the proposed online D2D offloading scheme is 38%, compared with the local execution scheme without offloading. Moreover, it is shown in 2(b) that the average delay per user of the proposed algorithm is much smaller than that of local execution scheme under any V. The delay reduction brought by online offloading becomes more obvious with the increase of V. Those comparisons demonstrate the advantage of proposed online D2D offloading scheme.

Fig. 3 reveals the impact of signaling overhead ratio α on both average network power consumption and average delay performance. The incentive factor is set to $\beta = 0.2$. As shown in Fig. 3(a), before the average network power consumption converging, i.e., $V < 4 \times 10^4$, the average network power consumption decreases slightly with the increases of α. Nevertheless, by increasing V from 4×10^4 to 10×10^4, the average power consumption of the proposed online offloading scheme with different α convergences to the same value, which is consistent with the Theorem 1 that the upper bound of the average power consumption is characterized by V and is independent of the α.

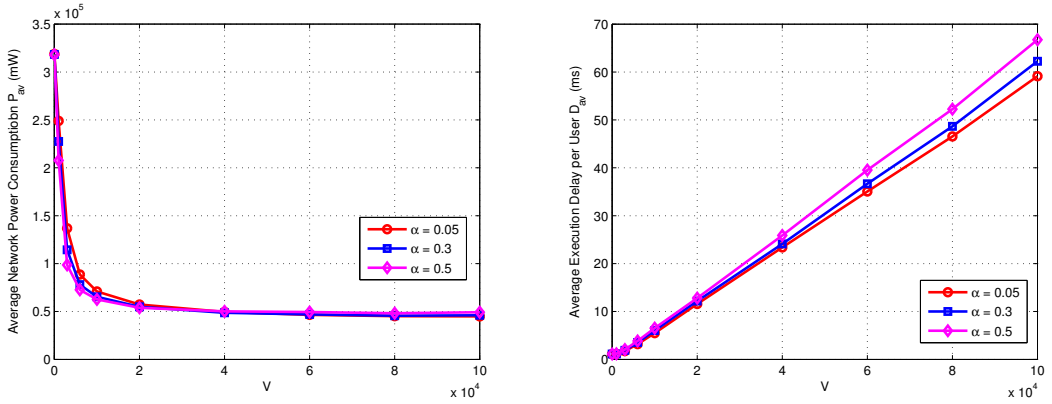
In Fig. 3(b), we demonstrate the impact of α on average delay performance. In general, the average delay increases with the increase of α under any V. Intuitively, larger signaling overhead ratio implies lower D2D offloading efficiency, and thus leads to high service delay. In addition, the average delay gap between different average delay performance becomes even lager with the increase of V. The reason behind this observation is that, according to the optimal solution for transmit power allocation (30), large V enlarges the impact of α on transmit power, thus leads to large delay gaps for different α.

In Fig.4, we investigate the relationship between average network power consumption and the average delay per user under different incentive factor β. The signaling overhead ratio α is set to $\alpha = 0.05$ and the is set to $a_i^{\max} = 5$ kbits/slot. In general, we can observe that in four situations, the service delay increases as the average network power consumption decreases, which further verifies the tradeoff between these two objects. In addition, given a proper V, the



(a) Average Network Power Consumption vs. Parameter V. (b) Average Delay per User vs. Parameter V.

Fig. 2: Comparison of online offloading scheme with local execution scheme.



(a) Average Network Power Consumption vs. Parameter V. (b) Average Delay per User vs. Parameter V.

Fig. 3: Impacts of signaling overhead ratio α on average network power consumption and average delay per user.

proposed algorithm with large incentive factor consumes less power and achieves smaller service delay. For example, when $V = 4 \times 10^4$, the average network power consumption of the proposed algorithm with $\beta = 0.8$ is 2.5×10^4 mW and the corresponding average delay is 11.1 ms, both of which are about half of that with $\beta = 0.2$. This is because, according to the definition of the incentive constraint, a larger β implies that the user is willing to contribute more resources in assisting others, and thus the performance improves.

By varying $a_i^{\max}$, we further explore the impacts of local task arrival rate on the average network power consumption and average delay performance in Fig. 5. Similar to Fig. 4, the average delay increases as the average network power consumption decreases. Besides, with a given control parameter V, both the average network power consumption and the average delay increase with the increase of task arrival rate. For instance, if the required average delay is 35 ms, the average power consumption with low task arrival ($a_i^{\max} = 3$ kbits) would be 1.1×10^4 mW while it is 1.2×10^5 mW with $a_i^{\max} = 7$ kbis/slot, which is about ten times than that with $a_i^{\max} = 3$ kbits. Those observations also conform with the intuition that, high workload certainly will leads to large execution latency and more power is needed to keep the task buffers stable.

Finally, we conduct the numerical studies regarding to the convergence time under the proposed algorithm by tracing the sum queue length of the computation task buffers in the network with different task arrival rate and control parameter V. Specifically, Fig. 6(a) illustrates the evolution of sum queue length under different task arrival rate. It is shown that queue backlogs for all three cases increase at the beginning and stabilizes around 100 kbits, 300 kbits and 700 kbits for the three cases respectively. Besides, Fig. 6(a) also demonstrates that high task arrival rate would lead to long convergence time. Fig. 6(b) depicts the the evolution of sum queue length under different control parameter V. It is shown that the stable queue backlogs increases with the increase of V, which again confirms that V indeed influences the average execution delay. Moreover, the convergence time slot also increases as the increase of V. Thus a proper V needs to be chosen in practice.

VI. Conclusion

In this paper, we investigated dynamic management of D2D task offloading and resource scheduling under the assistance of a centralized network controller in a fog-enabled D2D network. An energy-efficient online algorithm base on Lyapunov

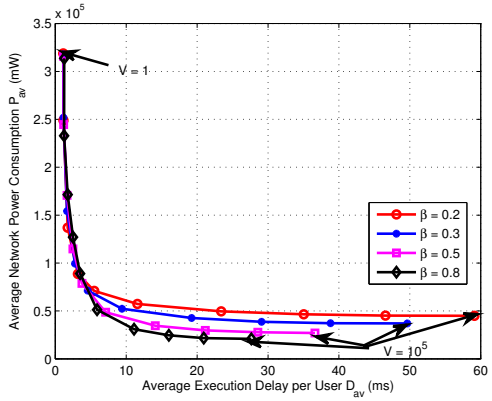


Fig. 4: Average Network Power Consumption vs. Average Delay per User under different incentive factor.

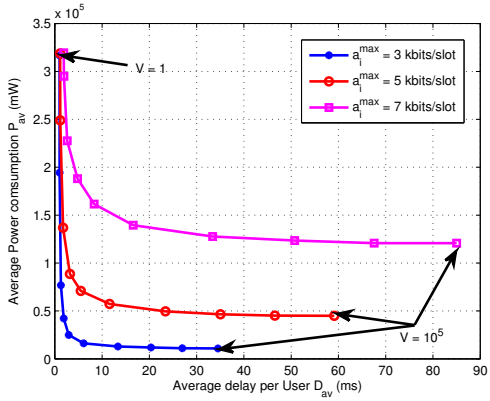
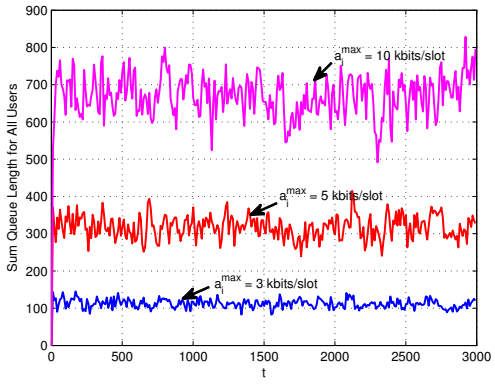
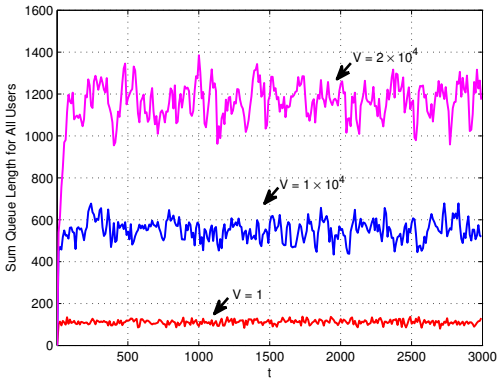


Fig. 5: Average Network Power Consumption vs. Average Delay per User under different task arrival rate.



(a) Network total queue backlog vs. t, $V = 6000$.



(b) Network total queue backlog vs. t, $a_i^{\max} = 5$ kbits/slot.

Fig. 6: Evolution of the sum queue length of the task buffers under the different parameters.

optimization was proposed, to minimize the average network power consumption and meanwhile ensuring the users' incentive and task buffer stability constraints. Performance analysis as well as simulations, explicitly characterize the tradeoff between the average power consumption and the average execution delay in the network. Furthermore, the impacts of various parameters were revealed, which confirms the benefit of online task offloading and provide valuable guidelines for the practical applications. For future it would be interesting to extend the findings in this work to scenarios with fairness considerations among the users.

VII. APPENDIX

A. Proof of Lemma 1

First, we use (22) to compute a bound on the change of Lyapunov drift function from one slot to the next:

$$
\begin{aligned}
&L(\mathbf{\Theta}(t+1)) - L(\mathbf{\Theta}(t)) = \frac{1}{2}\sum_{i\in\mathcal{U}}\left[Q_i^{\text{sum}}(t+1)^2 - Q_i^{\text{sum}}(t)^2\right] \\
&+\frac{1}{2}\sum_{i\in\mathcal{U}}\left[Z_i(t+1)^2 - Z_i(t)^2\right] \\
&= \frac{1}{2}\sum_{i\in\mathcal{U}}\left\{\left(\max\left[Q_i^{\text{sum}}(t) - C_i(t), 0\right] + A_i(t)\right)^2 - Q_i^{\text{sum}}(t)^2\right\} \\
&+\frac{1}{2}\sum_{i\in\mathcal{U}}\left\{\left(\max\left[Z_i(t) - \beta\mu_i^l(t), 0\right] + \mu_i^d(t)\right)^2 - Z_i(t)^2\right\} \\
&\leq \frac{1}{2}\sum_{i\in\mathcal{U}}\left(A_i(t)^2 + C_i(t)^2 + \mu_i^d(t)^2 + \beta^2\mu_i^l(t)^2\right) \\
&+\sum_{i\in\mathcal{U}} Q_i^{\text{sum}}(t)[A_i(t) - C_i(t)] + \sum_{i\in\mathcal{U}} Z_i(t)[\mu_i^d(t) - \beta\mu_i^l(t)],
\end{aligned}
\tag{34}
$$

where in the final inequality we have used the fact that for any $Q \geq 0, b \geq 0, A \geq 0$, we have:

$$(\max[Q-b,0]+A)^2 \leq Q^2+A^2+b^2+2Q(A-b). \quad (35)$$

Taking the conditional expectation of (34) and adding on both sides the penalty term $V\sum_{i\in\mathcal{U}}\mathbb{E}\{P_i(t)|\mathbf{\Theta}(t)\}$, we prove the *Lemma 1*.

B. Proof of Lemma 2

Proof: First, we define set $\mathcal{E}_i(t)=\{(i,j)|j\in\mathcal{U}\backslash i,(i,j)\in\mathcal{E}(t)\}$, which contains the possible D2D task offloading links for user i in time slot t, along with $\mathcal{E}^j(t)=\{(i,j)|i\in\mathcal{U}\backslash j,(i,j)\in\mathcal{E}(t)\}$, which consists of the possible D2D task arrival links for user j in time slot t. Next, the problem $\mathcal{P}_{\text{ua}}$ can be rewritten as below:

$$\max_{\underline{\mathcal{E}}(t)\in\mathcal{I}(t)}\sum_{(i,j)\in\underline{\mathcal{E}}(t)}W_{ij}(t), \quad (36)$$

where the family sets $\mathcal{I}(t)$ includes each subset $\underline{\mathcal{E}}(t)$ of $\mathcal{E}(t)$, such that the subset meets the constraint of $\boldsymbol{x}(t)$. Specifically,

$$\mathcal{I}(t)=\left\{\underline{\mathcal{E}}(t)\subseteq\mathcal{E}(t)\left|\begin{array}{l}|\underline{\mathcal{E}}(t)\cap\mathcal{E}_i(t)|\leq 1,\forall i;\\ |\underline{\mathcal{E}}(t)\cap\mathcal{E}^j(t)|\leq M,\forall j\end{array}\right.\right\}. \quad (37)$$

Invoking the definition of matroid in [23], we see that $\mathcal{I}(t)$ is the intersection of two partition matroids.

Then, with the definition of submodular function also given in [23], for all $\underline{\mathcal{E}}(t)\subseteq\underline{\mathcal{E}}'(t)\subseteq\mathcal{E}(t)$ and $(i',j')\in\mathcal{E}(t)\backslash\mathcal{E}'(t)$, we can deduce that the following relationship

$$\begin{aligned}&\sum_{(i,j)\in\{\underline{\mathcal{E}}(t)\cup(i',j')\}}W_{ij}(t)-\sum_{(i,j)\in\underline{\mathcal{E}}(t)}W_{ij}(t)\\ =&\sum_{(i,j)\in\{\underline{\mathcal{E}}'(t)\cup(i',j')\}}W_{ij}(t)-\sum_{(i,j)\in\underline{\mathcal{E}}'(t)}W_{ij}(t), \quad (38)\end{aligned}$$

and $\sum_{(i,j)\in\underline{\mathcal{E}}(t)}W_{ij}(t)=0$ if $\underline{\mathcal{E}}(t)=\emptyset$, which proves the desired result that $\mathcal{P}_{\text{ua}}$ can be formulated as the maximization of a normalized modular function with two partition matroid constraints.

C. Proof of Theorem 1

We first prove the second part (33). Denote the proposed online task offloading and resource scheduling algorithm as $\mathcal{O}^*(t)$, which achieves the suboptimal of the problem $\mathcal{P}1$ and also meets the Slater Type Condition (**??**). By applying the Slater Type Condition (**??**) into the right-hand-side of (25) and arranging the terms, we have:

$$\Delta_V(\mathbf{\Theta}(t))\leq\mathcal{K}+V\Psi_\epsilon-\delta\epsilon\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^{\text{sum}}(t)|\mathbf{\Theta}(t)\}. \quad (39)$$

By taking expectations on (39) and using the law of iterated expectations, we have:

$$\begin{aligned}&\mathbb{E}\{L(\mathbf{\Theta}(\tau+1))\}-\mathbb{E}\{L(\mathbf{\Theta}(\tau))\}+V\sum_{i\in\mathcal{U}}\mathbb{E}\{P_i(t)\}\\ &\leq\mathcal{K}+V\Psi_\epsilon-\delta\epsilon\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^{\text{sum}}(t)\}. \quad (40)\end{aligned}$$

By summing over $\tau\in\{0,1,\cdots,t-1\}$ for some slot $t>0$ and using the law of telescoping sums, we have:

$$\begin{aligned}&\mathbb{E}\{L(\mathbf{\Theta}(t))\}-\mathbb{E}\{L(\mathbf{\Theta}(0))\}-V\sum_{\tau=0}^{t-1}\sum_{i\in\mathcal{U}}\mathbb{E}\{P_i(t)\}\\ &\leq\mathcal{K}t+V\Psi_\epsilon t-\delta\epsilon\sum_{\tau=0}^{t-1}\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^{\text{sum}}(\tau)\}. \quad (41)\end{aligned}$$

By dividing both sides of (41) with $t\epsilon$, rearranging terms and using the fact that $\mathbb{E}\{L(\mathbf{\Theta}(t))\}>0$, we obtain:

$$\frac{1}{t}\sum_{\tau=0}^{t-1}\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^{\text{sum}}(\tau)\}\leq\frac{\mathcal{K}+V(\Psi_\epsilon-P^{\text{opt}})}{\delta\epsilon}. \quad (42)$$

By applying the definition of average traffic delay (14), We can deduce the average delay bound (33).

Next, we proceed to prove the upper bound of average power consumption (32). Note that the term $\delta\epsilon\sum_{i\in\mathcal{U}}\mathbb{E}\{Q_i^{\text{sum}}(t)|\mathbf{\Theta}(t)\}>0$, thus we can deduce:

$$\Delta(\mathbf{\Theta}(t))+V\sum_{i\in\mathcal{U}}\mathbb{E}\{P_i(t)\}\leq\mathcal{K}+V\Psi_\epsilon. \quad (43)$$

By summing (43) over $\tau\in\{0,1,\cdots,t-1\}$ and dividing it with Vt, and using the fact that $\mathbb{E}\{L(\mathbf{\Theta}(t))\}>0$, the average power consumption bound (32) can be proved.

D. Definitions of Submodular and Matroid

Submodular fuctions: Consider a finite ground set $\mathcal{E}$, and let $2^{|\mathcal{E}|}$ be the power set of $\mathcal{E}$. Then a function $f:2^{|\mathcal{E}|}\to\mathbb{R}$ is *submodular function* if and only if it has the property of diminishing returns: For all $\underline{\mathcal{E}}\subseteq\underline{\mathcal{E}}'\subseteq\mathcal{E}$ and $x\in\mathcal{E}\backslash\underline{\mathcal{E}}'$, we have

$$f(\underline{\mathcal{E}}\cup x)-f(\underline{\mathcal{E}})\geq f(\underline{\mathcal{E}}'\cup x)-f(\underline{\mathcal{E}}'), \quad (44)$$

and f is also called modular function if the above inequality becomes equality relationship.

Matroid: A *matroid* $\mathcal{S}$ must belong to $\{A\subseteq\mathcal{E}:|A|\leq k\}$, where k is a constant.

Partition matroid: Let the ground set $\mathcal{E}$ partition into disjoint sets $\mathcal{E}_1,\mathcal{E}_2,\cdots,\mathcal{E}_l$ and let $k_1,k_2,\cdots,k_l$ be positive integers. Then the matroid $\mathcal{S}$ is also a *partition matroid* when $\{\mathcal{S}\subseteq\mathcal{E}:|\mathcal{S}\cap\mathcal{E}_i|\leq k_i\}$.

REFERENCES

[1] S. Zhao, Y. Yang, X. Yang, X. Luo, and H. Qian, "Online user association and computation offloading for fog-enabled d2d network," in *2017 IEEE Fog World Congress (FWC)*, October 2017, pp. 1–6.

[2] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Computers and Communications (ISCC), 2012 IEEE Symposium on*, 2012, pp. 59–66.

[3] S. Tachi, M. Inami, and Y. Uema, "Augmented reality helps drivers see around blind spots," *IEEE Spectrum*, vol. 31, 2014.

[4] K. Kumar, J. Liu, Y.-H. Lu, and B. Bhargava, "A survey of computation offloading for mobile systems," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 129–140, 2013.

[5] S. Barbarossa, S. Sardellitti, and P. Di Lorenzo, "Communicating while computing: Distributed mobile cloud computing over 5G heterogeneous networks," *IEEE Signal Processing Magazine*, vol. 31, no. 6, pp. 45–55, 2014.

[6] L. Yang, J. Cao, H. Cheng, and Y. Ji, "Multi-user computation partitioning for latency sensitive mobile cloud applications," *IEEE Transactions on Computers*, vol. 64, no. 8, pp. 2253–2266, 2015.

[7] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless communications and mobile computing*, vol. 13, no. 18, pp. 1587–1611, 2013.

[8] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, 2010.

[9] J. Kwak, Y. Kim, J. Lee, and S. Chong, "Dream: Dynamic resource and task allocation for energy minimization in mobile cloud systems," *IEEE Journal on Selected Areas in Communications*, vol. 33, no. 12, pp. 2510–2523, 2015.

[10] L. M. Vaquero and L. Rodero-Merino, "Finding your way in the fog: Towards a comprehensive definition of fog computing," *ACM SIGCOMM Computer Communication Review*, vol. 44, no. 5, pp. 27–32, 2014.

[11] N. Chen, Y. Yang, T. Zhang, X. Luo, and J. Zao, "FA2ST: Fog as a service technology," *In submission*, 2017.

[12] X. Chen and J. Zhang, "When D2D meets cloud: Hybrid mobile task offloadings in fog computing," in *2017 IEEE International Conference on Communications (ICC)*, May 2017, pp. 1–6.

[13] Y. Mao, J. Zhang, S. Song, and K. B. Letaief, "Stochastic joint radio and computational resource management for multi-user mobile-edge computing systems," *arXiv preprint arXiv:1702.00892*, 2017.

[14] M. Xiao, J. Wu, L. Huang, Y. Wang, and C. Liu, "Multi-task assignment for crowdsensing in mobile social networks," in *2015 IEEE Conference on Computer Communications (INFOCOM)*, April 2015, pp. 2227–2235.

[15] L. Pu, X. Chen, J. Xu, and X. Fu, "D2d fogging: An energy-efficient and incentive-aware task offloading framework via network-assisted D2D collaboration," *IEEE Journal on Selected Areas in Communications*, vol. 34, no. 12, pp. 3887–3901, 2016.

[16] Y. Mao, J. Zhang, S. H. Song, and K. B. Letaief, "Power-delay tradeoff in multi-user mobile-edge computing systems," in *2016 IEEE Global Communications Conference (GLOBECOM)*, Dec 2016, pp. 1–6.

[17] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," in *Technologies for wireless computing*. Springer, 1996, pp. 119–137.

[18] H.-L. Fu, P. Lin, and Y.-B. Lin, "Reducing signaling overhead for femtocell/macrocell networks," *IEEE Transactions on Mobile Computing*, vol. 12, no. 8, pp. 1587–1597, 2013.

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

[20] S. M. Ross, *Introduction to probability models*. Academic press, 2014.

[21] L. Grippo and M. Sciandrone, "On the convergence of the block nonlinear gauss–seidel method under convex constraints," *Operations research letters*, vol. 26, no. 3, pp. 127–136, 2000.

[22] M. Porcelli and F. Rinaldi, "A variable fixing version of the two-block nonlinear constrained gauss–seidel algorithm for ℓ_1-regularized least-squares," *Computational Optimization and Applications*, vol. 59, no. 3, pp. 565–589, 2014.

[23] J. Edmonds, "Matroids and the greedy algorithm," *Mathematical programming*, vol. 1, no. 1, pp. 127–136, 1971.