Energy Delay Trade-Off in Cloud Offloading for Mutli-Core Mobile Devices

Article *in* IEEE Access · January 2015

DOI: 10.1109/ACCESS.2015.2499300

CITATIONS
76

READS
348

2 authors:

Zhefeng Jiang
Auburn University
9 PUBLICATIONS **149** CITATIONS
SEE PROFILE

Shiwen Mao
Auburn University
372 PUBLICATIONS **10,800** CITATIONS
SEE PROFILE

Received October 15, 2015, accepted October 28, 2015, date of publication November 10, 2015, date of current version November 24, 2015.

Digital Object Identifier 10.1109/ACCESS.2015.2499300

Energy Delay Tradeoff in Cloud Offloading for Multi-Core Mobile Devices

ZHEFENG JIANG, (Student Member, IEEE), AND SHIWEN MAO, (Senior Member, IEEE)
Department of Electrical and Computer Engineering, Auburn University, Auburn, AL 36849-5201, USA

Corresponding author: S. Mao (smao@ieee.org)

This work was supported in part by the National Science Foundation under Grant CNS-1247955 and in part by the Wireless Engineering Research and Education Center, Auburn University.

ABSTRACT Cloud offloading is considered a promising approach for both energy conservation and storage/computation enhancement for resource-limited mobile devices. In this paper, we present a Lyapunov optimization-based scheme for cloud offloading scheduling, as well as download scheduling for cloud execution output, for multiple applications running in a mobile device with a multi-core CPU. We derive an online algorithm and prove performance bounds for the proposed algorithm with respect to average power consumption and average queue length, which is indicative of delay, and reveal the fundamental tradeoff between the two optimization goals. The performance of the proposed online scheduling scheme is validated with trace-driven simulations.

INDEX TERMS Cloud computing, energy efficiency, job scheduling, Lyapunov optimization, offloading.

I. INTRODUCTION

There is a proliferation of mobile devices in recent years, such as smartphones and tablets, which are becoming more and more powerful with even multi-core CPUs. However, mobile devices still suffer from comparably limited resources. For example, the power of a smartphone comes at the cost of higher burden on the battery. As a result, although we are freed from a wireline data connection, we are still highly dependent on a power socket and charger. In addition, smartphones usually have relatively limited storage. With many apps, photos, and multimedia files recorded or cached, the internal storage space of our mobile devices can be easily depleted.

Cloud offloading has been recognized as an effective solution to the limited resource problem [2], [3]. With offloading, we can store our photos and videos in the cloud and fetch it whenever it is needed. Furthermore, computation intensive tasks can also be offloaded to software clones in the cloud [5], so that most computation can be executed in the cloud to greatly reduce the burden on the mobile device [6]. However, offloading data and computational tasks could involve considerable communications between mobile devices and cloud clones, which could consume a large amount of energy and incur extra delay. Hence, the decision between cloud offloading or local execution should be carefully made at each mobile device, taking into account the energy consumption and delay of various options, as well as the status of the wireless network.

In this paper, we study the problem of effective cloud offloading scheduling while considering downloading the output of cloud execution, for mobile devices with muti-core CPUs. We also consider task scheduling among the multiple cores of the CPU and frequency adaptation for the CPU, considering both energy cost and user experience with respect to delay. Specifically, there are several trade-offs in making the optimal decisions. First, cloud offloading involves data transmissions from the mobile device to the cloud, as well as downloading the output of cloud execution, through a stochastic and thus unpredictable wireless channel. The energy efficiency of cloud offloading could be poor when the wireless coverage is weak. In such cases, energy may be conserved if we delay cloud offloading and downloading until the channel gets better, but at the cost of additional delays. Furthermore, cloud offloading may not be a good choice for applications with a large amount of offloading data to be sent to the cloud, or a large amount of output data to be downloaded after cloud execution, since transmitting the data over a wireless channel may consume considerable power and incur large delay as well, which offset the gains achieved by executing the task in the cloud. Similarly, energy can be conserved for local execution by reducing the CPU frequency, but at the cost of slower execution (and thus increased delay) of the tasks.

Motivated by these observations, we present a holistic formulation of the problem of optimal cloud offloading decision making for multiple applications running in a multi-core

mobile device. The formulation takes into account the above trade-offs by incorporating the key control knobs, including CPU frequency and computation capability at the mobile device, offloading and downloading data volume of the applications, and the time-varying capacity and expected offloading power consumption of the wireless connection.

We then develop an effective solution algorithm to the formulated problem. The proposed scheduling algorithm is based on the Lyapunov optimizing framework [7], [8], [13]. It dynamically schedules the tasks in the task queues for cloud offloading or local execution, downloads output from the cloud for offloaded tasks, and in the case of local execution, tunes the CPU frequency to balance energy consumption and delay, based on the current network condition and task queue backlogs. The proposed algorithm is inherently an *online algorithm*, meaning that it does not require information about the stationary distributions of the arrival and wireless channel processes, neither does any future application and network state information. It makes decisions based on the current queue backlogs and wireless channel conditions. Such an online algorithm would be useful for real-time applications. We derive upper bounds on the average energy consumption and average queue length achieved by the proposed algorithm, which clearly reveal the trade-off between energy consumption and delay in optimal cloud offloading. The proposed algorithm is validated with trace-driven simulations, where the mobile device has both LTE and WiFi connections, and the energy-delay trade-off is clearly revealed.

The rest of this paper is organized as follows. The system model and problem statement are presented in Section II. The proposed algorithm is developed in Section III and evaluated with trace-driven simulations in Section IV. We review related work in Section V. Section VI concludes the paper. The main notation used in this paper is summarized in Table 1.

TABLE 1. Notation.

Symbol	Description
N	number of applications
$\mathcal{N}$	set of applications
N'	number of applications can be offloaded
$\mathcal{N}'$	set of applications can be offloaded
$\mathcal{Q}(t)$	set of application queues
$Q_i(t)$	queue of application i
$A_i(t)$	new arrivals to queue i at time t
$\mathcal{A}(t)$	set of arrivals at time t
λ_i	arrival rate of application i
$\vec{\lambda}$	set of arrival rate
$B_i(t)$	number of tasks of application i executed locally at time t
$B_i^O(t)$	number of executed tasks of application i downloaded at time slot t
$B_i^D(t)$	service rate of the cloud output queue for application i at time slot t
$\theta_i(k)$	computational complexity of task k of application i
$D_i(k)$	data size for offloading task k of application i
$D_i^D(k)$	data size of cloud execution output of task k of application i
$Q_i^D(t)$	returned output queue at of application i at the end of time slot t
$\mathcal{Q}^D(t)$	set of returned output queue at of application at the end of time slot t
$A_i^D(t)$	arrival to queue $Q_i^D(t)$ at time slot t
$\mathcal{A}_i(t)$	set of arrival to queues $Q^D(t)$ at time slot t
$f(t)$	clock frequency of CPU at time slot t
v	voltage of the mobile CPU at time t
η'	energy coefficient of CPU
$\varepsilon_i(t)$	energy consumption of core i at time slot t
$\varepsilon(t)$	overall energy consumption of the CPU at time t
$\alpha^L(t)$	set of application being executed locally
$\Theta_i(t)$	amount of computations a CPU core can offer to application i
M	number of CPU core
η	adjusted energy coefficient
$\omega_O(t)$	uplink wireless data rate
$\omega_D(t)$	downlink wireless data rate
$\alpha^O(t)$	offloaded application at time t
$p_O(t)$	energy consumption of offloading
$\alpha^D(t)$	application downloaded for execution data at time t
$p_D(t)$	energy consumption of downloading at time t
$\bar{P}$	average overall power consumption
$P(t)$	overall power consumption at time slot t
$\bar{Q}$	average overall queue length, including task queues and downloading queues
$L(Q(t))$	Lyapunov function
V_p	Lyapunov constant
P^{opt}	optimum (minimum) energy consumption
$\epsilon > 0$	distance between the data arrival rate vector $\vec{\lambda}$ and the system capacity region under the proposed algorithm
ξ	defined in (35), (37) and (38)
φ	a term defined in (29)

II. SYSTEM MODEL AND PROBLEM STATEMENT

A. SYSTEM MODEL

The system model is illustrated in Fig. 1. We consider a mobile device having N applications running,[1] denoted as $\mathcal{N} = \{1, 2, \cdots, N\}$, among which $1 \leq N' \leq N$ applications, denoted as $\mathcal{N}'$, can be offloaded to the cloud. The tasks generated from each application are enqueued and processed in a First-In-First-Out (FIFO) manner. In addition, we assume that the arrival and execution of these tasks follow a discrete, time-slotted system. In addition, most tasks can be processed in one slot, while very large tasks can be divided to smaller subtasks that can be processed in one time slot. In particular, the queue of tasks waiting to be processed for application i at the beginning of time slot t is denoted as $Q_i(t)$, and the overall queue lengths at the beginning of time slot t are denoted as

$$\mathcal{Q}(t) = \{Q_1(t), Q_2(t), \cdots, Q_N(t)\}. \tag{1}$$

[1] A multiple-thread application that enables parallel computing, can be treated as multiple applications.

In time slot t, the tasks generated by applications are denoted as

$$\mathcal{A}(t) = \{A_1(t), A_2(t), \cdots, A_N(t)\}, \tag{2}$$

which can be regarded as new arrivals to $\mathcal{Q}(t)$. In this paper, we assume that each $A_i(t)$ is independent and identically distributed (i.i.d.) over time slots and the expectations of them, i.e., the average arrival rates, are denoted as

$$\vec{\lambda} \triangleq \mathbb{E}\{\mathcal{A}(t)\} = \{\lambda_1, \lambda_2, \cdots, \lambda_N\}. \tag{3}$$

The departing tasks from queue $\mathcal{Q}(t)$ at time slot t is either scheduled for local execution, denoted as

$$\mathcal{B}(t) = \{B_1(t), B_2(t), \cdots, B_N(t)\}, \tag{4}$$

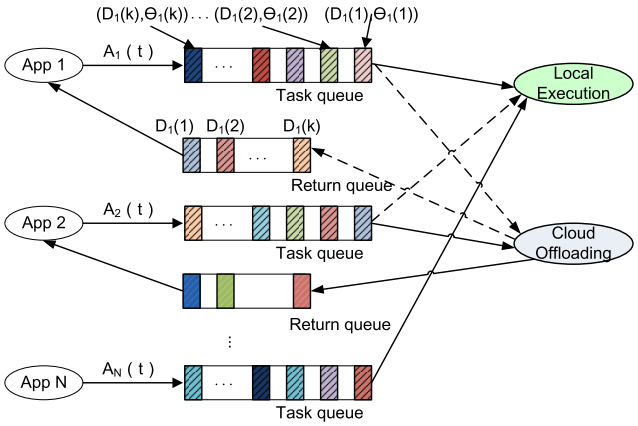

FIGURE 1. The system model.

or offloaded to the cloud, denoted as

$$\mathcal{B}^O(t) = \{B_1^O(t), B_2^O(t), \cdots, B_N^O(t)\}. \quad (5)$$

In addition, we assume that for task k of application i, the computational complexity for local execution, $\theta_i(k)$ (i.e., the amount of computations required to accomplish the task), the data size for offloading, $D_i(k)$ (i.e., the amount of data transmitted for executing the task in the cloud), and the data size of the cloud execution output, $D_i^D(k)$ (i.e., the results to be returned to the mobile device), are all i.i.d. random variables. If the task cannot be offloaded to the cloud, then we have $D_i(k) = \infty$ and $D_i^D(k) = 0$. Alternatively, if the task can only be offloaded to the cloud, then we have $\theta_i(k) = \infty$.

When a task is offloaded, it is first processed by a server in the cloud and then the output of cloud execution is returned to the mobile device. Hence, there is also a queue for the output data of cloud execution (e.g., at the access point or base station). Let $\mathcal{Q}^D(t)$ denote the returned output queue at the end of the time slot t, as shown in Fig. 1. We have

$$\mathcal{Q}^D(t) = \{Q_1^D(t), Q_2^D(t), \cdots, Q_N^D(t)\}, \quad (6)$$

where $Q_i^D(t) = 0$ for $i \in \mathcal{N}\backslash\mathcal{N}'$, as there will be no output from cloud computing if the task cannot be offloaded. The arrival to the queue $\mathcal{Q}^D(t)$ can be denoted as

$$\mathcal{A}^D(t) = \{A_1^D(t), A_2^D(t), \cdots, A_N^D(t)\}, \quad (7)$$

for an application i task that is to be offloaded, $|A_i^D(t)| = |B_i^O(t)|$. That is, if we ignore the time a cloud server takes to process the task, there is an increment of queue length in $Q_i^D(t)$ if a task in $Q_i(t)$ is offloaded to the cloud.

B. LOCAL EXECUTION ENERGY CONSUMPTION MODEL

For applications that are executed locally at the mobile device, most of the energy consumption comes from the CPU and the screen. As the screen energy consumption is largely dependent on the user habit, we do not take this part into account in this paper.[2] The energy consumption is thus mainly determined by the CPU operation.

[2]It may be annoying to dynamically adjust the display size, resolution, or brightness during the execution of an application. We simply assume some constant amount of energy consumption associated with this part.

In particular, the CPU energy consumption is proportional to v^2, where v is the CPU voltage [9]. Furthermore, the clock frequency of the CPU at time slot t, denoted as $f(t)$, is shown approximately linear to the CPU voltage v [9]. Therefore, the CPU power consumption in a CPU core occupied by application i in time slot t can be approximated as

$$\varepsilon_i(t) = \eta' \cdot f_i^2(t), \quad (8)$$

where η' is the energy coefficient determined by the CPU hardware architecture. As the energy consumption is linear with $f^2(t)$, energy can be saved by reducing the CPU frequency, which, however, will slow down the execution of the tasks.

A CPU schedule can be represented by $\{\alpha^L(t), \Theta(t)\}$, where $\alpha^L(t) \in \mathcal{N}$ is the set of applications being executed locally, $\Theta(t) = \{\Theta_1(t), \Theta_2(t), \cdots, \Theta_N(t)\}$, and $\Theta_i(t)$ is the amount of computations a CPU core can offer to application i at time slot t. Note that $\Theta_i(t) = 0$, if $i \notin \alpha^L(t)$. Assuming that there are M cores in the CPU. We have $|\alpha^L(t)| \leq M$, i.e., the number of parallel computing applications cannot exceed the number of cores in the CPU. For a given CPU architecture, the computational capability $\Theta_i(t)$ is usually linear with the CPU frequency. Hence, the CPU energy consumption at time t is also a quadratic function of $\Theta_i(t)$, i.e.,

$$\varepsilon_i(t) = \eta \cdot \Theta_i^2(t), \quad (9)$$

where η is the adjusted energy coefficient. The total energy consumption for local execution is

$$\varepsilon(t) = \sum_{i=1}^{N} \varepsilon_i(t). \quad (10)$$

C. OFFLOADING ENERGY CONSUMPTION MODEL

For applications that can be offloaded to the cloud, we make the following assumptions. First, we assume that a software clone has already been associated with each application in the cloud to support cloud computing [10], such that only the latest use generated data, application status updates, and cloud execution output, refereed to as *offloading data*, need to be transmitted between the mobile device and the cloud.

Second, we focus on the channel models associated with the wireless interfaces and ignore the delay and energy consumption in the cloud, which are justifiably minor issues comparing to that on the mobile device side. It is typical for a smartphone to choose one of the mobile networks (e.g., 2G, 3G, LTE, and WiFi) and the corresponding data rate is determined by the operator and the baseband chip configuration. We adopt the network selection algorithm proposed in [11] to choose between a cellular network and WiFi, and focus on the task scheduling problem in this paper.

Let $\omega_O(t)$ be the wireless link data rate from the mobile device to the cloud, and $\omega_D(t)$ the data rate from the cloud to the mobile device. An offloading decision is denoted as

$$\alpha^O(t) \in \{\mathcal{N}', '\text{idle}'\}. \quad (11)$$

That is, the device can choose to offload a task from one of the eligible queues or remain idle (i.e., to choose local execution). Then, the expected energy consumption is denoted as $p_O(t)$. Similarly, the decision for downloading the cloud execution output can be denoted as

$$\alpha^D(t) \in \{\mathcal{N}', ' \text{idle}'\}, \tag{12}$$

and the expected energy consumption is denoted as $p_D(t)$.

D. QUEUING AND THE OVERALL ENERGY CONSUMPTION MODEL

As discussed, energy can be conserved by optimizing the execution decision for the application tasks, i.e., local execution or offloading to the cloud. For local execution, energy can be saved by reducing the CPU frequency (i.e., running the application at a lower speed, which leads to a smaller $\Theta(t)$). For offloading, energy can be saved by only using good channels for transmission of offloading data and receiving the cloud output. There maybe an additional delay to wait for the channel to get better. If we aggressively save power by these means, the applications will suffer from large delays; the lengths of the task queues may increase to very high levels and the system may become unstable. We need to balance energy saving and delay, which is indicated by the task queue length.

Define the total power consumption in time slot t as

$$P(t) = \varepsilon(t) + p_O(t) + p_D(t). \tag{13}$$

Based on the local execution and offloading energy consumption models, the overall energy consumption of the mobile device can be derived as follows.

$$\begin{aligned}\bar{P} &\triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{P(t)\} \\ &= \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{\varepsilon(t) + p_O(t) + p_D(t)\}.\end{aligned} \tag{14}$$

We define the average task and output queue length, denoted as $\bar{Q}$, for evaluation of the energy-queue trade-off as

$$\bar{Q} \triangleq \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\sum_{i=1}^{N} \mathbb{E}\{Q_i(t) + Q_i^D(t)\}, \tag{15}$$

where $Q_i(t)$ is the task queue length for application i at time t, and $Q_i^D(t)$ is the cloud output queue length for application i at time t. We consider the system to be stable if the average queue length is bounded, i.e., the limit in (15) exists.

The dynamics of the task queue backlog $Q_i(t)$ can be written as

$$Q_i(t+1) = \max\{Q_i(t) + A_i(t) - B_i(t) - B_i^O(t), 0\}, \quad \forall i, \tag{16}$$

where $B_i(t)$ is the service rate at time t defined as follows.[3]

$$B_i(t) = \begin{cases} \arg\max_{\{b\}} \left\{\sum_{k=1}^{b} \theta_i(k) \le \Theta_i(t)\right\}, \\ \qquad\qquad\qquad \text{if } i \in \alpha^L(t) \\ \arg\max_{\{b\}} \left\{\sum_{k=1}^{b} D_i(k) \le \omega_O(t)\right\}, \\ \qquad\qquad\qquad \text{if } i \in \alpha^O(t) \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{17}$$

Note that $\alpha^L(t)$ and $\alpha^O(t)$ should not point to the same application i, as it is inefficient to both offload and locally execute the same application task at the same time. If $i \in \alpha^L(t)$, the task queue of application i is executed locally and $B_i(t)$ is the maximum number of tasks can be executed locally at time slot t. If $i \in \alpha^O(t)$, the tasks of application i are offloaded to the cloud and $B_i(t)$ is the maximum number of tasks can be offloaded at this time slot.

Similarly, the dynamics of the cloud output queue backlog $Q_i^D(t)$ can be written as

$$Q_i^D(t+1) = \max\{Q_i^D(t) + A_i^D(t) - B_i^D(t), 0\}, \quad \forall i \in \mathcal{N}', \tag{18}$$

where $\left|A_i^D(t)\right| = \left|B_i^O(t)\right|$ and $B_i^D(t)$ is the service rate at time i for the cloud output queue defined as

$$B_i^D(t) = \begin{cases} \arg\max_{\{b\}} \left\{\sum_{k=1}^{b} D_i^D(k) \le \omega_D(t)\right\}, \\ \qquad\qquad\qquad \text{if } i \in \alpha^D(t) \\ 0, \qquad\qquad\qquad \text{otherwise.} \end{cases} \tag{19}$$

If $i \in \alpha^D(t)$, the cloud output queue i is downloaded and $B^D(t)$ is the maximum number of tasks that can download their cloud output at this time slot.

E. PROBLEM STATEMENT

For a mobile device, it makes task scheduling decisions about offloading and local execution at the beginning of each slot. It then makes decisions for downloading the return data of cloud execution for the next slot at the end of current time slot. The objective of mobile devices is to keep all the queues stable and to minimize the overall energy consumption. The scheduling problem can be formulated as

$$\min: \ \limsup_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{\varepsilon(t) + p_O(t) + p_D(t)\} \tag{20}$$

$$\text{s.t. } \alpha^L(t) \cap \alpha^O(t) = \emptyset, \ \text{for all } t \tag{21}$$

$$\left|\alpha^L(t)\right| \le M, \ \text{for all } t \tag{22}$$

$$\bar{Q} < \infty, \tag{23}$$

where Constraint (21) forbids a task to be both executed locally and offloaded to the cloud in the same time slot, Constraint (22) is the limitation forced by the number of

[3] We assume that the duration of a time slot is large enough such that any task can be executed locally, offloaded to the cloud, or with output downloaded from the cloud in less than one time slot. This can be achieved by choosing a suitable time slot duration or by partitioning big tasks into smaller ones.

cores in the CPU, and Constraint (23) ensures stability of the task and output queues. The optimal solution to the problem consists of cloud offloading or local execution decisions for each time slot t (i.e., $\alpha^L(t)$ and $\alpha^O(t)$) and the optimized CPU computation capability $\Theta(t)$ for each time slot t, which translates to the optimal CPU clock frequency f as discussed in Section II-B (configured as in (37)).

III. TASK SCHEDULING ALGORITHM FOR MOBILE USERS

In this section, we present a task scheduling algorithm based on the Lyapunov optimization framework [7]. This algorithm requires no information about the stationary distributions of the arrival and wireless channel processes; it only requires information on the current queue lengths and the current channel conditions. Such an *online algorithm* property is useful for real-time applications [8], [12], [13].

A. LYAPUNOV OPTIMIZATION BASED SOLUTION ALGORITHM

To present the proposed algorithm, we first define a Lyapunov function $L(Q(t))$ as in [7].

$$L(Q(t)) \triangleq \frac{1}{2}\sum_{i=1}^{N} Q_i^2(t) + \frac{1}{2}\sum_{i=1}^{N}\{Q_i^D(t)\}^2, \tag{24}$$

where $L(Q(0)) = 0$. If all the queue lengths are small, then $L(Q(t))$ will be small; if at least one queue is congested, then $L(Q(t))$ will become large. Since there is a finite number of applications running on the mobile device, $L(Q(t))$ being bounded is equivalent to the notion that the system is stable.

Since $L(Q(0)) = 0$, for $L(Q(t+1))$, we have

$$\mathbb{E}\{L(Q(t+1))\} = \mathbb{E}\left\{\sum_{k=0}^{t}[L(Q(k+1)) - L(Q(k))]\right\}$$

$$= \sum_{k=0}^{t}\mathbb{E}\{L(Q(k+1)) - L(Q(k))|Q(k)\} = \sum_{k=0}^{t}\Delta(L(k)),$$

where $\Delta(L(t))$ is the *drift* defined as [14]

$$\Delta(L(t)) \triangleq \mathbb{E}\{L(Q(t+1)) - L(Q(t))\,|Q(t)\}. \tag{25}$$

We can minimize $\Delta(L(t))$ to maintain a low expectation for $L(Q(t))$. It follows (16) that

$$Q_i^2(t+1) \le \{Q_i(t) + A_i(t) - B_i(t) - B_i^O(t)\}^2. \tag{26}$$

For $i \in \alpha^O(t)$, we have

$$\{Q_i^D(t+1)\}^2 \le \{Q_i^D(t) + B_i^O(t) - B_i^D(t)\}^2. \tag{27}$$

For $i \notin \alpha^O(t)$, we have

$$\{Q_i^D(t+1)\}^2 \le \{Q_i^D(t) - B_i^D(t)\}^2. \tag{28}$$

Substituting (26), (27), and (28) into (25), we derive the drift (29) as given in the bottom of this page. In (29), φ denotes the terms in the expectation operators and

$$\Phi = \frac{1}{2}\sum_{i=1}^{N}\mathbb{E}\left\{\{A_i(t) - B_i(t) - B_i^O(t)\}^2 + \{B_i^O(t) - B_i^D(t)\}^2\right\}. \tag{30}$$

Note that $B_i^O(t) = 0$ for $i \notin \alpha^O(t)$. If the arrival rate and service rate of each queue is bounded, which is true for stable systems, then Φ is bounded.

As in [7], we obtain the *drift-plus-penalty*, defined as $\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\}\}$, by scaling the energy consumption with a positive coefficient V_p. The parameter V_p indicates the user's emphasis on energy consumption. Different applications can choose different V_p's to meet their specific delay requirements. Following (29), the upper bound of the *drift-plus-penalty* can be derived as

$$\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\} \le \Phi + \mathbb{E}\left\{\varphi + V_p \cdot P(t)\right\}. \tag{31}$$

To minimize the drift-plus-penalty, we can instead minimize $\{\varphi + V_p \cdot P(t)\}$ at every time slot, which only requires the current information on queue lengths, channel conditions, and the price for offloading.

Since there are M cores in the CPU of the mobile device, only M application can be executed by the CPU in each time slot. We assume that only one application can be offloaded at each time slot (through the single active wireless connection). We can derive the minimization expression as given in (32) on top of the next page. The first term in (32), $\sum_{i=1}^{N} Q_i(t)A_i(t)$, only depends on the current queue lengths and arrival rates. It does not affect the offloading downloading decision for this time slot. We need to minimize the second term

$$H_1 = V_p p_D(t) - Q_i^D(t)B_i^D(t)|_{i\in\alpha^D(t)}, \tag{33}$$

as a function of $\alpha^D(t)$, and the third term

$$H_2 = V_p\varepsilon(t) - \sum_{i\in\alpha^L(t)} Q_i(t)B_i(t) + \{V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\}, \tag{34}$$

as a function of $\alpha^L(t)$, $\alpha^O(t)$, and $\Theta(t)$.

Notice that for $\min\{H_1\}$, with the expectation of power consumption and offloading data, we need to find a

$$\Delta(L(t)) \le \Phi + \mathbb{E}\left\{\sum_{i\notin\alpha^O(t)} Q_i(t)(A_i(t) - B_i(t)) - Q_i^D(t)B_i^D(t)\right\} + \mathbb{E}\left\{\{Q_i(t)A_i(t) - (Q_i(t) - Q_i^D(t))B_i(t) - Q_i^D(t)B_i^D(t)\}|_{\{i\in\alpha^O(t)\}}\right\} = \Phi + \mathbb{E}\{\varphi\}. \tag{29}$$

$$\begin{aligned}
&\min\{\varphi + V_p P(t)\} = \min\left\{\sum_{i=1}^{N} Q_i(t)A_i(t) - \sum_{i=1}^{N} Q_i^D(t)B_i^D(t) - \sum_{i\notin\alpha^O(t)} Q_i(t)B_i(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)} + V_p P(t)\right\}\\
&= \sum_{i=1}^{N} Q_i(t)A_i(t) + \min\left\{V_p p_D(t) - \sum_{i=1}^{N} Q_i^D(t)B_i^D(t) + V_p\varepsilon(t) - \sum_{i\notin\alpha^O(t)} Q_i(t)B_i(t) + V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\right\}\\
&= \sum_{i=1}^{N} Q_i(t)A_i(t) + \min\left\{V_p p_D(t) - Q_i^D(t)B_i^D(t)|_{i\in\alpha^D(t)}\right\} + \min\left\{V_p\varepsilon(t) - \sum_{i\in\alpha^L(t)} Q_i(t)B_i(t) + \{V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\}\right\}\\
&= \sum_{i=1}^{N} Q_i(t)A_i(t) + \min\{H_1\} + \min\{H_2\}. \qquad (32)
\end{aligned}$$

proper $\alpha^D(t)$ that minimizes $-Q_i^D(t)B_i^D(t)$ in order to minimize the following function.

$$\xi_i^D = V_p p_D(t) - Q_i^D(t)B_i^D(t). \qquad (35)$$

This can be done by evaluating (35) for every application in $\mathcal{N}'$ to find the application i having the smallest ξ_i^D. Recall that $B_i^D(t)$ is defined in (19). For a given downlink capacity $\omega_D(t)$, tasks with smaller data size and longer queue length tend to have a smaller $-Q_i^D(t)B_i^D(t)$. Note that $V_p p_D(t) - Q_i^D(t)B_i^D(t)|_{i\in\alpha^D(t)} = 0$ when $\alpha^D(t) = 'idle'$. Thus a task will be offloaded in time slot t only when $\min\{V_p p_D(t) - Q_i^D(t)B_i^D(t)\} < 0$, meaning the channel condition is good or at least one of the task queues is long.

For the other term H_2, we need to minimize it by tuning $\alpha^L(t)$, $\alpha^O(t)$, and$\Theta(t)$. The term $V_p\varepsilon_i(t) - Q_i(t)B_i(t)$ can be rewritten as

$$\begin{aligned}
&V_p\varepsilon_i(t) - Q_i(t)B_i(t)\\
&= V_p\eta\Theta_i^2(t) - Q_i(t)\cdot\arg\max_{\{b\}}\left\{\sum_{k=1}^{b}\theta_i(k,t) \le \Theta_i(t)\right\}\\
&\cong V_p\eta\Theta_i^2(t) - Q_i(t)\frac{\Theta_i(t)}{\bar{\theta}_i(t)}, \qquad (36)
\end{aligned}$$

where $\bar{\theta}_i(t) = \frac{1}{Q_i(t)}\sum_{k=1}^{Q_i(t)}\theta_i(k,t)$. We can derive the approximate minimum value $V_p\varepsilon(t) - Q_i(t)B_i(t)$ subject to the CPU computation capability $\Theta_i(t)$ as

$$\xi_i^L(t) = -\frac{Q_i^2(t)}{4V_p\eta\bar{\theta}_i^2(t)}, \text{ when } \Theta_i(t) = \frac{Q_i(t)}{2V_p\eta\bar{\theta}_i(t)}. \qquad (37)$$

Similarly, we can evaluate (37) for all the applications in $\mathcal{N}$ and find the minimizer. Since the computational capability of the CPU cannot be increased indefinitely, we set an upper bound for the CPU power, e.g., 10 W in this paper.

For the term $\{V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\}$, we can minimize it by tuning $\alpha^O(t)$. Denoting

$$\xi_i^O = V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t) < 0, \qquad (38)$$

an application $i \in \mathcal{N}'$ with smaller offloading data size and greater $Q_i(t) - Q_i^D(t)$ will achieve a smaller ξ_i^O. Also note that $\{V_p p_O(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\} = 0$ when $\alpha_i^O =' idle'$. Thus a task can be offloaded only when $\xi_i^O < 0$.

Then the $\min\{H_2\}$ term can be rewrite as

$$\min\{H_2\} = \min\left\{\sum_{i\in\alpha^L(t)} \xi_i^L + \xi_j^O|_{j\in\alpha^O(t),\alpha^O(t)\cap\alpha^L(t)=\emptyset}\right\}.$$

According to the above evaluation, the problem becomes

$$\begin{aligned}
&\sum_{i=1}^{N} Q_i(t)A_i(t) + \min\{H_1\} + \min\{H_2\}\\
&= \sum_{i=1}^{N} Q_i(t)A_i(t) + \min\left\{\xi_i^D\right\}\\
&\quad + \min\left\{\sum_{i\in\alpha^L(t)} \xi_i^L + \xi_j^O|_{j\in\alpha^O(t),\alpha^O(t)\cap\alpha^L(t)=\emptyset}\right\}, \qquad (39)
\end{aligned}$$

where ξ_i^D, ξ_i^L, and ξ_j^O are defined in (35), (37) and (38), respectively. We also have $\alpha^O(t)\cap\alpha^L(t) = \emptyset$, since the same application cannot be executed locally and offloaded to the cloud in the same time slot. The proposed task scheduling algorithm is presented in Algorithm 1, where all computations except Step 2 are simple operations.

Algorithm 1: Task Scheduling Algorithm

```
Update all the task queues and estimate wireless link capacities at the beginning of time slot t ;
Find the minimum combination of ∑_{i∈α^L(t)} ξ_i^L + ξ_j^O, where α^O(t) ∩ α^L(t) = ∅ and j ∈ N' ;
if ξ_j^O < 0 then
    Offload tasks of application j to the cloud ;
end
for i ∈ α^L(t) do
    if ξ_i^L < 0 then
        Execute tasks of application i locally, with CPU capacity Θ_i(t) = Q_i(t) / (2V_p η θ̄_i(t)) ;
    end
end
Find the minimum ξ_i^D;
if ξ_i^D < 0 then
    Fetch the output data for application i tasks from the cloud ;
end
```

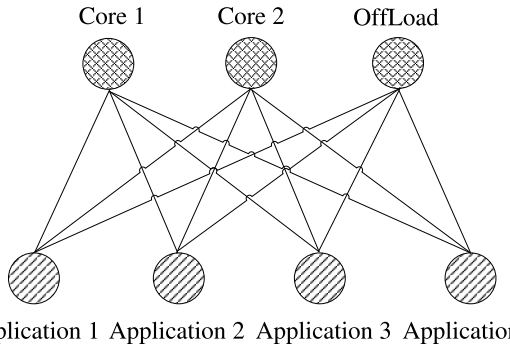

FIGURE 2. Task scheduling as a minimum weighted matching of a bipartite graph (illustrated for $N = 4$ and $M = 2$).

For Step 2 in Algorithm 1, the task scheduling can be illustrated as a minimum weighted matching of a bipartite graph as shown in Fig. 2. In the graph, vertex Application i, $i = 1, 2, \cdots, N$ represent the applications, vertex Core i, $i = 1, 2, \cdots, L$ stands for the cores in the CPU, and vertex OffLoad stands for the offloading link. The edge between vertice Application i and Core j means that it can be executed locally on core j and the weight of the edge is ξ_i^L. Correspondingly, the edge between vertice Application i and OffLoad means that it can be offloaded to cloud, while the weight of the edge is ξ_i^O. In Step 2, we need to find the selection edges with minimum weight, and according to constraint (21) and (22), each vertex can only be connected with one selected edge. Then it is a maximum weighted bipartite matching problem and can be solved with Hungarian algorithm [15] with complexity $O(N * (M+1)^2)$ if $(M + 1 < N)$, or $O((M + 1) * N^2)$ otherwise.

In Algorithm 1, at the beginning of each time slot t, the mobile device first updates the queues of tasks and estimates the capacity of wireless capacities to compute ξ_i^L, ξ_i^O, and ξ_i^D. In Step 2, it finds out smallest combination of $\sum_{i\in\alpha^L(t)} \xi_i^L + \xi_j^O$, where $\alpha^O(t) \cap \alpha^L(t) = \emptyset$, since a task should not be computed locally and offloaded to cloud at the same time. Then it offloads the corresponding task of application j if $\xi_j^O < 0$ and computes the tasks of application $i \in \alpha^L(t)$ if $\xi_i^L < 0$, with $\Theta_i(t) = \frac{Q_i(t)}{2V_p\eta\tilde{\Theta}_i(t)}$. At last, the mobile user makes the decision on downloading the output of cloud computing. It first finds the smallest ξ_i^D for all applications in $\mathcal{N}'$. If $\xi_i^D < 0$ for the smallest ξ_i^D, then it downloads the corresponding output of cloud computing.

B. PERFORMANCE ANALYSIS

Following the framework of Lyapunov optimization [7], we derive the upper bounds for the expected average power consumption and the expected average queue length achieved by the proposed algorithm, which are summarized in the following theorem. The proof is presented in the Appendix.

Theorem 1: Assume that the arrival rate of tasks $\vec{\lambda}$ is strictly within the system capacity region. That is, the system can maintain stability under certain $\{\alpha^L(t), \alpha^O(t), \alpha^D(t), \Theta(t)\}$. *Then the bounds on average energy consumption and queue length under Algorithm 1 can be written as*

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T-1} E\{P(t)\} \le P^{opt} + \frac{\Phi}{V_p} \quad (40)$$

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T-1} \sum_{i=1}^{N} E\{Q_i(t) + Q_i^D(t)\} \le \frac{1}{\epsilon}(\Phi + V_p P), \quad (41)$$

where P^{opt} is the minimum energy consumption a stable system can achieve, P is the average energy consumption under the proposed algorithm, and $\epsilon > 0$ is the distance between the data arrival rate vector $\vec{\lambda}$ and the system capacity region under the proposed algorithm.

Theorem 1 demonstrates the trade-off between energy consumption and queue length (or, delay). The upper bound of the average energy consumption is $O(1/V_p)$ and the upper bound of the average queue length is $O(V_p)$. Therefore these are conflicting objectives. We can tune V_p to flexibly trade off between energy consumption and queue length. When the power supply is not so limited (e.g., a charger is available), the user can decrease V_p to reduce the queue length (and thus delay) and enjoy better quality of experience (QoE). On the other hand, if the power constraint is stringent (e.g., the mobile device is running out of battery and no charger is available), the user can increase V_p to save energy at the expense of longer average queue length and larger delay.

IV. TRACE-DRIVEN SIMULATION VALIDATION

We evaluate the performance of the proposed algorithm with trace-driven simulations. In the simulations, we adopt the wireless network measurement data gathered by testing the data rate of the LTE/WiFi networks while walking around the Auburn University campus with an iPhone5. The LTE carrier is AT&T and the WiFi network is deployed by Auburn university. In particular, half of the LTE rate tests are conducted outdoor and half of the tests are conducted indoor. The WiFi rate tests are conducted in Broun Hall in the Auburn University Campus.

In the simulations, the wireless link data rate is randomly selected from the measured trace. For power consumption, we adopt the power models for LTE and WiFi proposed in [16]. For the uplink, the LTE power model can be approximated as $p_O = a_{LTE} \cdot \omega_O + b_{LTE}$, where $a_{LTE} = 0.5$ W, $b_{LTE} = 1.25$ W, and ω_O is the wireless network data rate in Mbps. For WiFi, the power consumption mode is $p = a_{WiFi} \cdot \omega_O + b_{WiFi}$, where $a_{WiFi} = 0.24$ W and $b_{WiFi} = 0.125$ W. For downlink, the power model for LTE can be approximated as $p_D = a_{LTE}^D \cdot \omega_D + b_{LTE}^D$, where $a_{LTE}^D = 0.042$ W, $b_{LTE}^D = 1.25$ W. For WiFi, the power consumption mode is $p_D = a_{WiFi}^D \cdot \omega_D + b_{WiFi}^D$, where $a_{WiFi}^D = 0.12$ W and $b_{WiFi}^D = 0.125$ W.

We consider a scenario with five applications running in the mobile device and all of them can be offload. The task arrival rate of each application ranges from 0.5 to 2.0.

The offloading data size of the tasks follows a truncated Exponential distribution with means ranging from 60 KB to 300 KB. For local execution, η was set to 0.6 corresponding to the normalized computation complexity Θ. The normalized computation complexity of each task follows an Exponential distribution with means ranging from 0.1 to 1. In the simulations, V_p is increased from 1 to 200. For each V_p value, the simulation runs for 50,000 time slots.

We compare the following four schemes in the simulations: (i) the proposed scheme with single core CPU, (ii) the proposed scheme with dual core CPU, (iii) the proposed scheme with single core CPU, and with Large Output of Cloud Computing (LOCC) (i.e. the average data size of cloud computing is twice of that of offloading), and (iv) the "eTime" strategy proposed in [14] with LOCC.

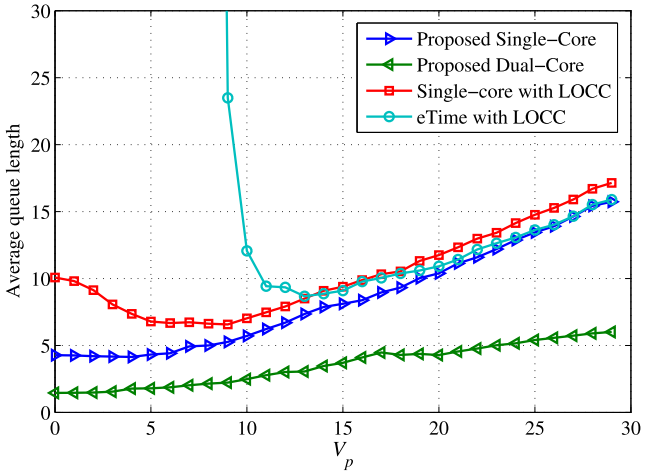

FIGURE 3. Average queue length of the four schemes.

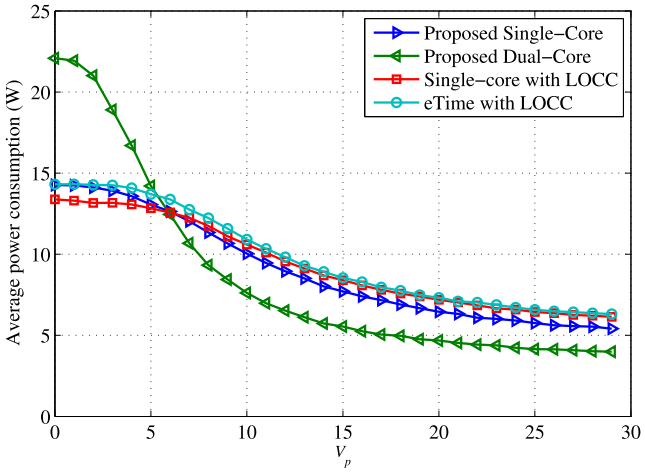

FIGURE 4. Average power consumption of the four schemes.

The simulation results are plotted in Figs. 3 and 4 for average queue length and average power consumption, respectively. It can be seen that there is a clear trade-off between average energy consumption and average queue length achieved by tuning V_p for both single core and dual core CPU. When V_p is increased, the average energy consumption is decreased but the average queue length is increased. It confirms the findings in Theorem 1 that the average queue length follows $O(V_p)$ (see Fig. 3) and the average energy consumption follows $O(1/V_p)$ (see Fig. 4) asymptotically. When V_p is smaller than 10, the energy consumption decreases rapidly with V_p, while the average queue length increases almost linearly with V_p. Therefore, users can achieve big energy savings, while only suffers a linearly increased delay, by increasing V_p in this region. From the simulation, we can find clearly that for dual core CPU, the queue length is much shorter than that of the single CPU system. But the power consumption for dual core is much higher with small V_p, but with high V_p (i.e., larger than 4), the system with dual core CPU enjoy lower energy consumption. It means that system with dual core system enhances the system computation ability and show greater flexibility for trade off between energy consumption and queue length.

For system with single core CPU with LOCC, it suffers from longer queue length and greater energy consumption with large V_p (i.e. greater than 4), as the downloading for Output of Cloud Computing is more resource consumption. The queue length of the single core CPU system with LOCC suffers from a high queue length with the low V_p(ie. smaller than 4), that is because the system offloading tasks aggressively with low V_p and the downloading for output of cloud is resource consuming, which increases the average queue length. With low V_p (i.e. smaller than 4), the power consumption of single core CPU system with LOCC consumes less energy consumption than that of single core CPU system. It is because that the single core CPU system with LOCC has longer queue for downloading the output of cloud computing, which results in a smaller ξ_j^O and effects of V_p is enhanced.

The simulation results also demonstrate that the performance of the proposed algorithm is better than that of the strategy proposed in [14] with LOCC, which suffers higher energy consumption. In addition, in the LOCC scenario, eTime couldn't stabilize the system with a low V_p. It is because that with a low V_p, eTime aggressively offloads tasks to the cloud but couldn't download the output of cloud execution.

V. RELATED WORK

Cloud offloading is regarded as an effective solution to save energy, extend storage spaces, and enable computation intensive applications at mobile devices [2]–[4]. There have been many prior work addressing the various design issues of cloud computing to fully harvest its potential [5], [17]–[20], [23], [24]. In particular, considerable recent works have focused on building the framework of enable mobile computation offloading [5], [20], [23], [24], suggesting for a mobile device to execute codes remotely in resource-rich servers, which connect the mobile device through LAN or wireless link. Ref. [24] implemented method level offloading for applications on Microsoft .NET, and Ref. [20] implemented

a flexible application partitioner which enables seamlessly offloading of part of the execution to the virtual machine. On the other hand, many other works [17], [25], [28] have focused on backing up data and applications to extend the storage space of mobile devices. However, both computation offloading and data/application backup involve considerable energy consumption for data transmission between mobile devices and the cloud, which may makes some excellent techniques [30] infeasible in practical implementation scenarios.

Researchers have started to investigate the energy cost of offloading [10], [14], [18], [21], [22], [24], [26]–[29], [31], [33]. Some techniques focused on reducing the energy consumption during offloading [21], [24], [27]–[29], [31]. For example, in [21], the authors proposed a dynamic offloading algorithm to save energy by offloading some components of an application to the cloud, while Ref. [31] proposed an algorithm to reduce energy consumption by selecting the most energy efficient WiFi AP for offloading. Furthermore, some researches have investigated the tradeoff between energy consumption and delay [10], [18], [22], [26], [32]. For example, the bandwidth and energy costs of cloud computing were investigated in [10]. In [26], a heuristic algorithm was proposed to jointly minimize the energy consumption and delay. However, these works are based on static models of application, and more important, the stochastic characteristics of applications and network dynamics have not been taken into consideration. The authors of [18] and [22] proposed an energy-optimal mobile computing framework under stochastic wireless channels, while considering the single application scenario. In [14], an energy-efficient transmission algorithm between the cloud and mobile devices was presented based on the Lyapunov optimizing framework [7]. However, the local computation resources in the mobile devices has not been fully utilized, and it doesn't consider downloading the cloud execution output.

This work was motivated by the above interesting works to investigate the energy-delay trade-off in cloud offloading with a Lyapunov optimization approach. We explicitly considered the stochastic nature of both user and application behaviors, and network dynamics, and addressed the more challenging case of multiple applications, thus greatly extending the work in [18] and [22]. This work also extended prior work [14] by considering multi-core CPUs and fully utilizing the local computing capability, by making offloading decisions based on both task queues and queues for downloading the output of cloud execution. As in [14], the online operation of the proposed scheme makes it highly suitable for real-time applications.

VI. CONCLUSIONS

In this paper, we proposed a scheduling scheme for energy-efficient cloud offloading for muti-core mobile devices, while considering downloading the cloud execution output in the model. Based on Lyapunov optimization, we developed an online algorithm that does not require information about stationary distribution of applications and the network condition, making it amenable to real-time implementation for practical scenarios. We proved theoretical bounds for the proposed algorithm and validated its performance with trace-driven simulations.

APPENDIX

A. PROOF OF THEOREM 1

According to (29) and (32), we have

$$\begin{aligned}
&\min\{\varphi + V_p P(t)\} \\
&= \min\Bigg\{V_p P(t) + \sum_{i=1}^{N} Q_i(t)A_i(t) - \sum_{i=1}^{N} Q_i^D(t)B_i^D(t) \\
&\quad - \sum_{i\notin\alpha^O(t)} Q_i(t)B_i(t) - (Q_i(t) - Q_i^D(t))B_i^O(t)|_{i\in\alpha^O(t)}\Bigg\} \\
&= \min\Bigg\{V_p P(t) + \sum_{i=1}^{N} Q_i^D(t)(B_i^O(t) - B_i^D(t)) \\
&\quad + \sum_{i=1}^{N} Q_i(t)(A_i(t) - B_i(t) - B_i^O(t))\Bigg\} \\
&\le V_p P^*(t) + \sum_{i=1}^{N} Q_i^D(t)(B_i^{*O}(t) - B_i^{*D}(t)) \\
&\quad + \sum_{i=1}^{N} Q_i(t)(A_i(t) - B_i^*(t) - B_i^{*O}(t)),
\end{aligned} \tag{42}$$

where $P^*(t)$, $B_i^*(t)$, $B_i^{*O}(t)$ and $B_i^{*D}(t)$ are the terms corresponding to any other (possibly randomized) feasible schedule. Now consider a randomized scheduling policy that achieves the following for Application $i \in \mathcal{N}$.

$$\mathbb{E}\{P^*(t)\} = P^{opt} \tag{43}$$

$$\mathbb{E}\left\{B_i^{*O}(t) - B_i^{*D}(t)\right\} \le 0 \tag{44}$$

$$\mathbb{E}\left\{A_i(t) - B_i^*(t) - B_i^{*O}(t)\right\} \le 0, \tag{45}$$

where P^{opt} is the minimum power consumption a stable system can achieve and (44) and (45) stabilize the queues.

For the proposed algorithm, we have

$$\begin{aligned}
&\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\} \\
&\le V_p \cdot \mathbb{E}\{P(t)\} + \Phi + \mathbb{E}\{\varphi\} \\
&\le V_p \cdot \mathbb{E}\{P^*(t)\} + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i^D(t)(B_i^{*O}(t) - B_i^{*D}(t))\right\} \\
&\quad + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i(t)(A_i(t) - B_i^*(t) - B_i^{*O}(t))\right\} + \Phi \\
&\le V_p \cdot P^{opt} + 0 + \Phi,
\end{aligned} \tag{46}$$

where

$$\mathbb{E}\left\{\sum_{i=1}^{N} Q_i^D(t)(B_i^{*O}(t) - B_i^{*D}(t))\right\} \le 0 \tag{47}$$

$$\mathbb{E}\left\{\sum_{i=1}^{N} Q_i(t)(A_i(t) - B_i^*(t) - B_i^{*O}(t))\right\} \leq 0, \quad (48)$$

according to (43) and (44).

Then we have

$$\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\} \leq V_p \cdot P^{opt} + \Phi, \quad (49)$$

and $\sum_{k=0}^{T-1} \Delta(L(t)) = L(T) < \infty$ for a stable system. It follows that

$$\begin{aligned}
&\limsup_{t\to\infty} \frac{1}{T}\sum_{k=0}^{T-1} \Delta(L(T)) + \limsup_{t\to\infty} \frac{V_p}{T}\sum_{k=0}^{T-1} \mathbb{E}\{P(t)\} \\
&= 0 + \limsup_{t\to\infty} \frac{V_p}{T}\sum_{k=0}^{T-1} \mathbb{E}\{P(t)\} \\
&\leq V_p \cdot P^{opt} + \Phi.
\end{aligned}$$

Then we have that (40) holds true.

Suppose for Application $i \in \mathcal{N}$, there exist some real number $\epsilon > 0$, such that

$$\mathbb{E}\left\{B_i^O(t) - B_i^D(t)\right\} \leq -\epsilon \quad (50)$$

$$\mathbb{E}\left\{A_i(t) - B_i(t) - B_i^O(t)\right\} \leq -\epsilon. \quad (51)$$

According to (42), we have

$$\begin{aligned}
&\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\} \\
&\leq V_p \cdot \mathbb{E}\{P(t)\} + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i^D(t)(B_i^O(t) - B_i^D(t))\right\} \\
&\quad + \mathbb{E}\left\{\sum_{i=1}^{N} Q_i(t)(A_i(t) - B_i(t) - B_i^O(t))\right\} + \Phi \\
&\leq V_p \cdot \mathbb{E}\{P(t)\} + \Phi - \epsilon \cdot \mathbb{E}\left\{\sum_{i=1}^{N}(Q_i(t) + Q_i^D(t))\right\}. \quad (52)
\end{aligned}$$

As $\sum_{i=0}^{T-1}\{\Delta(L(t)) + V_p \cdot \mathbb{E}\{P(t)\}\} \geq 0$, we have

$$\mathbb{E}\left\{\sum_{i=1}^{N}(Q_i(t) + Q_i^D(t))\right\} \leq \frac{1}{\epsilon}\{V_p \cdot \mathbb{E}\{P(t)\} + \Phi\}. \quad (53)$$

It follows that

$$\begin{aligned}
&\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T-1}\sum_{i=1}^{N} E\{Q_i(t) + Q_i^D(t)\} \quad (54) \\
&\leq \frac{\Phi}{\epsilon} + \frac{1}{\epsilon}\limsup_{T\to\infty} \frac{1}{T}\sum_{i=0}^{T-1}\{V_p \cdot \{P(t)\}\} \\
&= \frac{1}{\epsilon}(\Phi + V_p P),
\end{aligned}$$

and we conclude that (41) holds true.

REFERENCES

[1] Z. Jiang and S. Mao, "Energy delay trade-off in cloud offloading for multi-core mobile devices," in *Proc. IEEE GLOBECOM*, San Diego, CA, USA, Dec. 2015, pp. 1–6.

[2] Y. Xu and S. Mao, "A survey of mobile cloud computing for rich media applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 46–53, Jun. 2013.

[3] Y. Xu and S. Mao, "Mobile cloud media: State of the art and outlook," in *Mobile Computing Over Cloud: Technologies, Services, and Applications*, J. Rodrigues, K. Lin, and J. Lloret, Eds. Hershey, PA, USA: IGI Global, 2013, pp. 18–38.

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

[5] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.

[6] M. Chen, Y. Zhang, Y. Li, S. Mao, and V. C. M. Leung, "EMC: Emotion-aware mobile cloud computing in 5G," *IEEE Netw.*, vol. 29, no. 2, pp. 32–38, Mar./Apr. 2015.

[7] M. J. Neely, *Stochastic Network Optimization With Application to Communication and Queueing Systems*. San Rafael, CA, USA: Morgan & Claypool Publishers, 2010.

[8] Y. Huang, S. Mao, and R. M. Nelms, "Adaptive electricity scheduling in microgrids," *IEEE Trans. Smart Grid*, vol. 5, no. 1, pp. 270–281, Jan. 2014.

[9] T. D. Burd and R. W. Brodersen, "Processor design for portable systems," *J. VLSI Signal Process. Syst.*, vol. 13, nos. 2–3, pp. 203–221, Aug./Sep. 1996.

[10] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 1285–1293.

[11] A. J. Nicholson, Y. Chawathe, M. Y. Chen, B. D. Noble, and D. Wetherall, "Improved access point selection," in *Proc. ACM MobiSys*, Uppsala, Sweden, Jun. 2006, pp. 233–245.

[12] Y. Wang, S. Mao, and R. M. Nelms, "Online algorithm for optimal real-time energy distribution in the smart grid," *IEEE Trans. Emerg. Topics Comput.*, vol. 1, no. 1, pp. 10–21, Jun. 2013.

[13] Z. Jiang and S. Mao, "Online channel assignment, transmission scheduling, and transmission mode selection in multi-channel full-duplex wireless LANs," in *Proc. WASA*, Qufu, China, Aug. 2015, pp. 243–252.

[14] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, and Y. Qu, "eTime: Energy-efficient transmission between cloud and mobile devices," in *Proc. IEEE INFOCOM*, Turin, Italy, Apr. 2013, pp. 195–199.

[15] H. W. Kuhn, "The Hungarian method for the assignment problem," *Naval Res. Logistics Quart.*, vol. 2, nos. 1–2, pp. 83–97, Mar. 1955. [Online]. Available: http://dx.doi.org/10.1002/nav.3800020109

[16] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4G LTE networks," in *Proc. ACM MobiSys*, Low Wood Bay, U.K., Jun. 2012, pp. 225–238.

[17] W. Zhang, Y. Wen, Z. Chen, and A. Khisti, "QoE-driven cache management for HTTP adaptive bit rate streaming over wireless networks," *IEEE Trans. Multimedia*, vol. 15, no. 6, pp. 1431–1445, Oct. 2013.

[18] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. O. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.

[19] Y. Jin, Y. Wen, H. Hu, and M.-J. Montpetit, "Reducing operational costs in cloud social TV: An opportunity for cloud cloning," *IEEE Trans. Multimedia*, vol. 16, no. 6, pp. 1739–1751, Oct. 2014.

[20] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. EuroSys*, Salzburg, Austria, Apr. 2011, pp. 301–314.

[21] D. Huang, P. Wang, and D. Niyato, "A dynamic offloading algorithm for mobile computing," *IEEE Trans. Wireless Commun.*, vol. 11, no. 6, pp. 1991–1995, Jun. 2012.

[22] Y. Wen, W. Zhang, and H. Luo, "Energy-optimal mobile application execution: Taming resource-poor mobile devices with cloud clones," in *Proc. IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 2716–2720.

[23] 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.

[24] E. Cuervo *et al.*, "MAUI: Making smartphones last longer with code offload," in *Proc. ACM MobiSys*, San Francisco, CA, USA, Mar. 2010, pp. 49–62.
[25] A. Leff and J. T. Rayfield, "Integrator: An architecture for an integrated cloud/on-premise data-service," in *Proc. IEEE ICWS*, New York, NY, USA, Jun./Jul. 2015, pp. 98–104.
[26] X. Wang, J. Wang, X. Wang, and X. Chen, "Energy and delay tradeoff for application offloading in mobile cloud computing," *IEEE Syst. J.*, vol. PP, no. 99, pp. 1–10, Aug. 2015.
[27] C. Xu, Y. Qiao, B. Lee, and N. Murray, "Energy consumption of mobile offloading for JavaScript applications," in *Proc. 26th Irish Signals Syst. Conf. (ISSC)*, Carlow, Republic of Ireland, Jun. 2015, pp. 1–6.
[28] Y.-S. Chen, C.-S. Hsu, T.-Y. Juang, and H.-H. Lin, "An energy-aware data offloading scheme in cloud radio access networks," in *Proc. IEEE WCNC*, New Orleans, LA, USA, Mar. 2015, pp. 1984–1989.
[29] Z. Chang, J. Gong, Z. Zhou, T. Ristaniemi, and Z. Niu, "Resource allocation and data offloading for energy efficiency in wireless power transfer enabled collaborative mobile clouds," in *Proc. IEEE INFOCOM WKSHPS*, Hong Kong, Apr./May 2015, pp. 336–341.
[30] W. Enck *et al.*, "TaintDroid: An information-flow tracking system for realtime privacy monitoring on smartphones," in *Proc. USENIX OSDI*, Vancouver, BC, Canada, Oct. 2010, pp. 1–6.
[31] A. Y. Ding *et al.*, "Enabling energy-aware collaborative mobile data offloading for smartphones," in *Proc. IEEE SECON*, New Orleans, LA, USA, Jun. 2013, pp. 487–495.
[32] K. Kumar and Y.-H. Lu, "Cloud computing for mobile users: Can offloading computation save energy?" *Computer*, vol. 43, no. 4, pp. 51–56, Apr. 2010.
[33] M. Altamimi, A. Abdrabou, K. Naik, and A. Nayak, "Energy cost models of smartphones for task offloading to the cloud," *IEEE Trans. Emerg. Topics Comput.*, vol. 3, no. 3, pp. 384–398, Sep. 2015.

ZHEFENG JIANG (S'15) received the bachelor's degree in electrical engineering from Beijing Jiaotong University, Beijing, China in 2009, and the master's degree in electrical engineering from Tsinghua University, Beijing, China in 2012. He is currently pursuing the Ph.D. degree with the Department of Electrical and Computer Engineering, Auburn University, Auburn, AL, USA. His research interests include cognitive radios, cloud computing, full-duplex wireless networks, and LTE in unlicensed bands.

SHIWEN MAO (S'99–M'04–SM'09) received the Ph.D. degree in electrical and computer engineering from Polytechnic University, Brooklyn, NY, in 2004. He is currently the Samuel Ginn Endowed Professor and the Director of the Wireless Engineering Research and Education Center with the Samuel Ginn College of Engineering, Auburn University, Auburn, AL, USA. His research interests include wireless networks and multimedia communications, such as cognitive radio, small cells, millimeter wave networks, free space optical networks, localization, and smart grid. He received the 2015 IEEE ComSoc Communication Switching and Routing Technical Committee (TC-CSR) Distinguished Service Award, the 2013 IEEE ComSoc Multimedia Communications Technical Committee (MMTC) Outstanding Leadership Award and the NSF CAREER Award in 2010. He was a co-recipient of the IEEE WCNC 2015 Best Paper Award, the IEEE ICC 2013 Best Paper Award, and the 2004 IEEE Communications Society Leonard G. Abraham Prize in the field of communications systems. He is a Distinguished Lecturer of the IEEE Vehicular Technology Society. He is on the Editorial Board of the IEEE TRANSACTIONS ON MULTIMEDIA, the IEEE INTERNET OF THINGS JOURNAL, the IEEE COMMUNICATIONS SURVEYS AND TUTORIALS, and the *IEEE Multimedia*. He serves as the Area TPC Chair of the IEEE INFOCOM 2016, the Technical Program Vice Chair for Information Systems of the IEEE INFOCOM 2015, the Symposium Co-Chair for many conferences, including the IEEE ICC, the IEEE GLOBECOM, ICCCN, and the IEEE ICIT-SSST, the Steering Committee Voting Member of the IEEE ICME and AdhocNets, and in various roles in the organizing committees of many conferences. He is the Vice Chair–Letters and a member of Communications of the IEEE ComSoc MMTC.