Dual-side Dynamic Controls for Cost Minimization in Mobile Cloud Computing Systems

Yeongjin Kim, Jeongho Kwak and Song Chong
Department of Electrical Engineering, KAIST
E-mail: {yj.kim, jh.kwak}@netsys.kaist.ac.kr, songchong@kaist.edu

***Abstract*—Mobile cloud computing (MCC) has been proposed to offload heavy computing jobs of mobile devices to cloud servers managed by cloud service provider (CSP), which enables the mobile devices to save energy and processing delay. Heretofore, cloud offloading policies in mobile devices and pricing/scheduling in CSP have been independently addressed. This paper is first to jointly account for both sides of mobile users and CSP in a unified mobile cloud computing framework. By invoking "Lyapunov drift-plus-penalty" technique, we propose dual-side control algorithms for the mobile users and CSP in two different scenarios: *(i)* In non-cooperation scenario, we propose a NC-UC algorithm for the mobile users and a NC-CC algorithm for the CSP to minimize each cost for given delay constraints. *(ii)* In cooperation scenario, we suggest a CP-JC algorithm for both cloud users and CSP to minimize the sum costs of them for given delay constraints. Trace-driven simulations demonstrate that NC-UC saves minimum 63% of cost by trading 8MB of average queue lengths when compared with the existing algorithms, and NC-CC achieves 71% of profit gain when compared with the same delay of existing scheme; moreover, the cooperation enables them to save additional costs and delays.**

I. Introduction

According to the forecasts of Cisco [1], 11.2EB of mobile data traffic will be generated in 2017, which is 7-folds higher than that in 2013, mainly due to high quality of the multimedia contents which also require high processing resources. To cope with such computation-intensive applications, the Mobile Cloud Computing (MCC) has been proposed as an emerging technology to offload the computing-intensive jobs (e.g., face recognition or video transcoding) in mobile devices to cloud service provider (CSP). The MCC is already widely utilized as a form of commercial cloud services, e.g., Windows Azure [2], Amazon EC2 [3] and Google cloud platform [4]. ABI research forecasts that more than 240 million mobile business users will use cloud services driving $5 billion in revenues by the end of 2015 [5].

The MCC has two kinds of benefits for mobile users: *(i)* It enables mobile devices to save local processing energy. Because the CPU power is drastically increasing to the increment of the clock speed [6], heavy processing jobs by local CPU is extremely big burden for the mobile devices. *(ii)* It enables the mobile devices to reduce the processing delay with the help of the cloud server of which the processing speed is much faster than that of mobile devices [7]. However, the mobile users may not obtain these advantages for free due to the networking energy to offload the computing jobs to the cloud server and the expense to use the MCC service. Therefore, the mobile users should carefully determine the cloud offloading or adjust local CPU clock frequency to minimize their costs for local computing (e.g., local processing energy) or cloud computing (e.g., networking energy and prices managed by the CSP for the cloud services) while satisfying the processing delay constraints. For example, let us assume that the achievable data rate of a mobile device is high, the cloud computing is low-priced and the processing density, defined as required CPU clock cycles to process one bit[1], of jobs is high. Then, the mobile user would prefer to offload the computing jobs to the cloud in order to reduce the user's costs.

On the other hand, the CSP is able to obtain the benefit from the mobile users for the MCC service. However, willingnesses to pay of mobile users are heterogeneous depending on their network states and the load of computing jobs; hence the pricing is a challenging problem. Also, they should make a payment for electricity usage which is time-varying [8] for operating cloud servers. Thus, the objective of the CSP is to determine price for the MCC service and adjust processing amount in cloud servers by selecting the active number of servers in order to maximize long-term benefits from mobile users minus payment for electricity usage while satisfying the processing delay constraints. For example, the CSP decides the price for the MCC service to earn more money with considering the willingness to pay of mobile users. Also, the CSP uses more active servers to process requested jobs from users when the electricity bill to operate cloud servers is inexpensive.

Previous studies related to the mobile cloud computing where the objective is to minimize energy consumption of *user-side* (mobile device) under static computing job size and static wireless channel state scenarios [9]–[11]. However, arrival of computing jobs and network states of real life are time-varying, those assumptions have limitations in practice. Also, those studies do not concern the cost of cloud computing which directly affects to the offloading decision of users. In most of previous studies dealing with user-side mobile

This work makes use of results produced by the SmartFIRE project, which is supported by the International Research & Development Program of the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP, Korea) (Grant number: K2013078191) and the Seventh Framework Programme (FP7) funded by the European Commission (Grant number: 611165).

[1]Each workload generally requires different cycles in processing the same amount of data in bits.

978-3-9018-8273-9/15/ ©2015 IFIP

cloud computing, they assume that the processing capacity of cloud server is infinite, so that they solve the problem without considering the processing delay at the cloud, which cannot capture the real environment in the cloud servers.

Other works studied cloud computing to minimize energy consumption [12] or to maximize profit [13], [14] of *cloud-side* (cloud data center). The weakness of those studies is that they did not strictly model the user-side such as arrival of computing job, offloading policy of user and willingness to pay for using the MCC service.

Although there have been several studies about MCC algorithms for user-side and cloud-side, respectively, most of them did not jointly optimize the sum costs of users and CSP which is also called social cost. If the users and CSP cooperate with each other to reduce social cost and distribute their costs by paying computing price between the users and CSP appropriately, both sides of them can expect additional gains in terms of costs. For example, a user does not transfer computing jobs to the cloud but process locally to reduce social cost even though offloading is favorable to the user. On the other side, the CSP process the requested computing job to reduce processing delay of overall system even though the cost for operating cloud server is quite high. One work [15] dealt with the cooperation scenario in terms of pricing among the users and CSP, however it does not capture real environment of MCC such as local computing strategy of users. More details about related works are summarized in a technical report [16].

In this paper, we propose dual-side dynamic controls[2] in mobile cloud computing systems in two scenarios: *(i)* Users and CSP do not cooperate with each other, i.e., they are trying to minimize their own cost for given delay constraints, respectively. *(ii)* Users and CSP cooperate with each other to minimize the social cost for given processing delay constraints. To exploit the opportunism of dynamic variations of wireless conditions, job arrivals and electricity bills, we invoke "Lyapunov drift-plus-penalty" technique [17] in both user-side and cloud-side that does not require information about the distribution of arrivals, network states and electricity bills, but only needs to know information about the current states of them. To model realistic cloud system environment, we consider finite processing capacity of cloud servers, real processing density of computing jobs, time-varying electricity bills for operating the cloud servers. We run trace-driven simulations over real measurements of LTE data rates and energy consumption of CPU and LTE interface in popular smartphone models. Our paper is first to concretely model the strategies of mobile users and CSP simultaneously and reveal cost-delay tradeoff of dual-side in mobile cloud systems.

The contributions of this paper are summarized as follows.

- We propose dual-side algorithms in MCC which minimize the cost of users by scheduling user-side resources (cloud offloading or local computing) and selecting CPU clock speed, called *NC-UC*, and minimize the cost of CSP by determining the active cloud servers and selecting the MCC price, called *NC-CC*, for non-cooperation scenario.
- We propose a control algorithm, called *CP-JC*, to minimize the social cost by jointly determined control parameters in dual-side, for cooperation scenario. This approach is the first to observe the social cost reduction by cooperation between the users and CSP.
- Through trace-driven simulation, we demonstrate that NC-UC saves 63% of cost by trading 8MB of average queue lengths, NC-CC achieves 71% of profit gain when compared with the same delay of existing schemes; moreover, the cooperation between cloud users and CSP enables them to achieve 22% of additional social cost reduction (10^4KB queue length) and 47% of additional delay reduction (1.1×10^{-3}\$ social cost).

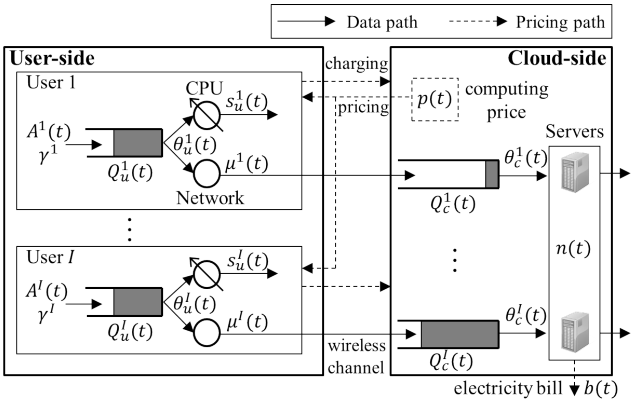


Fig. 1: Framework for mobile cloud computing system

In the rest of our paper, we describe system model in Section II. In Section III, we propose two kinds of dual-side control algorithms. Next, in Section IV, we evaluate our algorithms by trace-driven simulations. Finally, we conclude our paper in Section V.

II. System Model

Job & arrival model. Fig. 1 illustrates our framework for a mobile cloud computing system. We consider one CSP and I number of mobile users who subscribe to the MCC service where $i \in \mathcal{I}$ is a user index. We assume a time-slotted system indexed by $t \in \{0, 1, ...\}$. At each time slot, $A^i(t)$ size (in bits) of computing job is arrived for user i and $A^i(t)$ is an independent and identically distributed process where $\mathbb{E}\left[A^i(t)\right] = \lambda^i$. We assume that all arrivals in every slot are bounded, i.e., $A^i(t) \leq A^i_{\max}$, for all $i \in \mathcal{I}$. Computing job of each user has a different processing density γ^i (in cycles/bit) which is defined as required CPU cycles to process a unit bit. We assume that all computing jobs can be separated into independent and fine-grained tasks, and they are delay tolerant [9][3]. User-side jobs can be served by two types of processing resources: *(i)* the local CPU resources in the mobile devices, *(ii)* the cloud resources in the CSP.

User-side resource model. Typically, modern smartphone processors have DVFS (Dynamic Voltage and Frequency

[2]We define dual-side control as controlling user-side and cloud-side simultaneously.

[3]They do not have instantaneous delay constraints.

Scaling) capability, so that it adjusts CPU clock speed, i.e., $s_u^i(t) \in \mathcal{S}_u$ (in cycles/time slot), every time slot where the set of adjustable CPU clock speeds is defined as $\mathcal{S}_u = \{0, s_{u,1}, s_{u,2}, \ldots, s_{u,\max}\}$, and $s_u^i(t) = 0$ means to stop processing the computing job and let CPU be in the idle state. For simplicity, we regard that all smartphones have the same sets of adjustable CPU clock speeds. Every time slot, the user i determines the local CPU clock speed $s_u^i(t)$, and computing resources between local CPU and cloud, which is denoted by $\theta_u^i(t)$ for all users as follows.

$$\theta_u^i(t) = \begin{cases} 1, \text{ if the job is transferred to the cloud server,} \\ 0, \text{ if the job is processed in the local CPU.} \end{cases}$$

Network speed of user i, $\mu^i(t) \in (0, \mu_{\max}^i]$, is given every time slot t. We assume that the user is always in cellular network coverage. We note that uplink data rates of the network technology are predictable enough by online and offline estimation exploiting received signal strength indicator (RSSI) and past data rates history [18].

Cloud-side resource model. The CSP has $n(t)$ number of available servers where $n(t) \in \{0, 1, ..., n_{\max}\}$, which changes every slot. We assume that cloud-side jobs of the corresponding mobile user can be served by at most only one server every time slot. The finite processing capacity of unit server is denoted by s_c (in cycles/slot). The CSP determines time-dependent pricing $p(t) \in [0, \infty)$ (in \$/cycle) for using the MCC service. Every time slot t, the CSP determines the cloud computing price $p(t)$, and selects jobs to process, which is denoted by $\boldsymbol{\theta}_c(t) = (\theta_c^1(t), \theta_c^2(t), ...\theta_c^I(t))$ as follows.

$$\theta_c^i(t) = \begin{cases} 1, \text{ if the job of user } i \text{ is selected,} \\ 0, \text{ otherwise.} \end{cases}$$

In order to capture the fact that the number of selected servers cannot exceed the number of total servers, we consider a constraint as follows.

$$\sum_{i \in \mathcal{I}} \theta_c^i(t) \leq n(t) \tag{1}$$

Queueing model. For each mobile user, we denote $Q_u^i(t)$ by a user-side queue length of computing job for user i (in bits). For the CSP, we consider *I* number of cloud-side queues for each subscriber. These queues are required due to the finite processing capacity of the cloud servers. Then, we denote $Q_c^i(t)$ by the cloud-side queue length (in bits) of user i at time slot t. We model queueing dynamics at the user-side and cloud-side as follows.

$$Q_u^i(t+1) = \left[Q_u^i(t) - (1-\theta_u^i(t))\frac{s_u^i(t)}{\gamma^i} - \theta_u^i(t)\mu^i(t) + A^i(t)\right]^+, \tag{2}$$

$$Q_c^i(t+1) = \left[Q_c^i(t) - \theta_c^i(t)\frac{s_c}{\gamma^i} + \theta_u^i(t)\mu^i(t)\right]^+, \tag{3}$$

where $[x]^+ = \max(x, 0)$. The amount of departure from the user-side queue is determined by the control parameters $(\theta_u^i(t), s_u^i(t))$. The amount of arrival to and departure from the server-side queue is determined by control parameters $\theta_u^i(t)$ and $\theta_c^i(t)$, respectively. Because the unit of $s_u^i(t)$ in the local CPU (and s_c in the cloud server) is cycles/time slot and that of the queue lengths is bits, the amount of served workloads from the queue is $s_u^i(t)$ (and s_c) divided by γ^i for unit agreement.

Cost model. Processing job through the local CPU resources of a mobile user requires the energy cost which is a function of CPU clock speed $E_C(s_u^i(t))$ (in Joule/time slot). On the other hand, if the computing job is served by cloud resources, two kinds of costs are required for the mobile user. *(i)* One is networking energy E_N (in Joule/time slot) in transferring computing job to the cloud. We assume that all mobile users exploit the same network technology, e.g., 3G or LTE and the transmit powers of their devices are known to the users [6]. *(ii)* The other is a pricing cost $p(t)\gamma^i\mu^i(t)$ which the user has to pay to CSP for using cloud service. We denote $b(t)$ by an electricity bill (in \$) required in activating one server during time slot t which is paid by the CSP.

Then, the costs (in \$) of user i and the CSP at time slot t are as follows.

$$\text{User } i\text{:} \quad h_u^i(t) = (1-\theta_u^i(t))\alpha^i(t)E_C(s_u^i(t)) + \theta_u^i(t)(\alpha^i(t)E_N + p(t)\gamma^i\mu^i(t)), \tag{4}$$

$$\text{CSP:} \quad h_c(t) = \sum_{i \in \mathcal{I}} \theta_c^i(t)b(t) - \sum_{i \in \mathcal{I}} \theta_u^i(t)\gamma^i\mu^i(t)p(t). \tag{5}$$

where $\alpha^i(t)$ (in \$/J) is a tradeoff parameter between the price for using MCC service and the energy consumed in the device of mobile user i at time slot t. This is a user-dependent factor decided by sensitiveness on price and energy. Furthermore, it can be changed over time according to charging status of mobile user. We assume that $\alpha^i(t)$ is not influenced by past decisions.

III. Dual-side Control Algorithm

In this section, we formulate two optimization problems considering the cost minimization with queuing stability for *(i)* non-cooperation case (i.e., the users and CSP try to minimize their own costs, respectively) and *(ii)* cooperation case (i.e., the users and CSP cooperate with each other to minimize their social cost). For non-cooperation case, we develop a user-side control algorithm called NC-UC, a cloud-side control algorithm, called NC-CC. For cooperation case, we develop a joint control algorithm, called CP-JC.

A. Problem Formulation

Our objectives and constraints for non-cooperation case of users, (NC-U), and CSP, (NC-C), and for cooperation case, (CP), in a mobile cloud computing system are summarized in Table I. Control parameters $(\boldsymbol{\theta}_u, \boldsymbol{s}_u, \boldsymbol{p}, \boldsymbol{\theta}_c)$ denote as follows.

$$\begin{cases} \boldsymbol{\theta}_u = (\theta_u^1(t), \theta_u^2(t), \ldots, \theta_u^I(t))_{t=0}^T, \\ \boldsymbol{s}_u = (s_u^1(t), s_u^2(t), \ldots, s_u^I(t))_{t=0}^T, \\ \boldsymbol{p} = (p(t))_{t=0}^T, \quad \boldsymbol{\theta}_c = (\theta_c^1(t), \theta_c^2(t), \ldots, \theta_c^I(t))_{t=0}^T. \end{cases}$$

The objective (NC-U) is to minimize the cost of user i by controlling $(\boldsymbol{\theta}_u^i, \boldsymbol{s}_u^i)$ for given CSP's policy. It is constrained by the queue stability of both user-side and cloud-side in order to process the jobs within finite time. The objective

TABLE I: Objectives and Constraints

	(NC-U)	(NC-C)	(CP)
Objective	$\min\limits_{(\boldsymbol{\theta}_u^i,\boldsymbol{s}_u^i)}\Big[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\Big\{h_u^i(t)\Big\}\Big], \forall i$	$\min\limits_{(\boldsymbol{p},\boldsymbol{\theta}_c)}\Big[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\Big\{h_c(t)\Big\}\Big],$	$\min\limits_{(\boldsymbol{\theta}_u,\boldsymbol{s}_u,\boldsymbol{\theta}_c)}\Big[\lim\limits_{T\to\infty}\frac{1}{T}\sum\limits_{t=0}^{T-1}\mathbb{E}\Big\{\sum\limits_{i\in\mathcal{I}}h_u^i(t)+h_c(t)\Big\}\Big],$
Constraints	$\limsup\limits_{t\to\infty}\frac{1}{t}\sum\limits_{\tau=0}^{t-1}\mathbb{E}\Big\{Q_u^i(\tau)+Q_c^i(\tau)\Big\}<\infty,$	$\limsup\limits_{t\to\infty}\frac{1}{t}\sum\limits_{\tau=0}^{t-1}\mathbb{E}\Big\{\sum\limits_{i\in\mathcal{I}}Q_c^i(\tau)\Big\}<\infty,$ $\sum_{i\in\mathcal{I}}\theta_c^i(t)\le n(t),\forall t,$	$\limsup\limits_{t\to\infty}\frac{1}{t}\sum\limits_{\tau=0}^{t-1}\mathbb{E}\Big\{\sum\limits_{i\in\mathcal{I}}\big(Q_u^i(\tau)+Q_c^i(\tau)\big)\Big\}<\infty,$ $\sum_{i\in\mathcal{I}}\theta_c^i(t)\le n(t),\forall t,$

(NC-C) is to minimize the cost of CSP by controlling $(\boldsymbol{p},\boldsymbol{\theta}_c)$ for given policy of the user. It is constrained by the queue stability of cloud-side because interest of the CSP is to process the received jobs from users within finite time. The objective (CP) is to minimize the social cost by controlling $(\boldsymbol{\theta}_u,\boldsymbol{s}_u,\boldsymbol{\theta}_c)$ with the queue stability constraint of user-side and cloud-side. Because the cloud computing price $\boldsymbol{p}$ does not affect the social cost (i.e., the benefits of CSP from users is exactly same as the payment of users to the CSP), we do not care $\boldsymbol{p}$ in the cooperation case.

B. Algorithm Design

We design the control algorithms for (NC-U), (NC-C) and (CP) by invoking "Lyapunov drift-plus-penalty" framework [17] which has advantages in the sense that it does not require information about the distributions of job arrivals, network states and electricity price, but only needs to know information about the current states of theirs.

Making slot-by-slot objective. First, we define Lyapunov function and Lyapunov drift function as follows.

$$\text{(NC-U): } L(t)=\frac{1}{2}\Big\{(Q_u^i(t))^2+(Q_c^i(t))^2\Big\} \tag{6}$$

$$\Delta L(t)=\mathbb{E}\big\{L(t+1)-L(t)\big|Q_u^i(t),Q_c^i(t)\big\} \tag{7}$$

$$\text{(NC-C): } L(t)=\frac{1}{2}\Big\{\sum_{i\in\mathcal{I}}(Q_c^i(t))^2\Big\} \tag{8}$$

$$\Delta L(t)=\mathbb{E}\big\{L(t+1)-L(t)\big|\boldsymbol{Q}_c(t)\big\} \tag{9}$$

$$\text{(CP): } L(t)=\frac{1}{2}\Big\{\sum_{i\in\mathcal{I}}\big((Q_u^i(t))^2+(Q_c^i(t))^2+(Q_u^i(t)+Q_c^i(t))^2\big)\Big\} \tag{10}$$

$$\Delta L(t)=\mathbb{E}\big\{L(t+1)-L(t)\big|\boldsymbol{Q}_u(t),\boldsymbol{Q}_c(t)\big\} \tag{11}$$

$$\text{where } \begin{cases}\boldsymbol{Q}_u(t)=(Q_u^1(t),Q_u^2(t),\ldots,Q_u^I(t)),\\ \boldsymbol{Q}_c(t)=(Q_c^1(t),Q_c^2(t),\ldots,Q_c^I(t)).\end{cases} \tag{12}$$

The Lyapunov functions (6), (8) and (10) are designed to fairly stabilize queues among users in terms of delay. In (6), $(Q_u^i(t))^2$ is trying to minimize the processing delay at the user-side and $(Q_c^i(t))^2$ is trying to avoid congestion at the cloud-side. The function (8) is trying to minimize the processing delay at the cloud-side. In (10), $(Q_u^i(t))^2$ and $(Q_c^i(t))^2$ are trying to minimize the processing delay at the user-side and cloud-side, repectively, and $(Q_u^i(t)+Q_c^i(t))^2$ means the CSP and user i try to minimize the processing delay of waiting jobs for user i because user i and the CSP cooperate with each other. Then, minimizing Lyapunov drifts (7), (9) and (12) for each objective function is the same as satisfying the constraints in Table I.

Next, we define "Lyapunov drift-plus penalty" functions where the penalty function is the cost during time slot t as follows.

$$\text{(NC-U): } \Delta L(t)+V\mathbb{E}\Big\{h_u^i(t)\Big|Q_u^i(t),Q_c^i(t)\Big\}, \tag{13}$$

$$\text{(NC-C): } \Delta L(t)+V\mathbb{E}\Big\{h_c(t)\Big|\boldsymbol{Q}_c(t)\Big\}, \tag{14}$$

$$\text{(CP): } \Delta L(t)+V\mathbb{E}\Big\{\sum_{i\in\mathcal{I}}h_u^i(t)+h_c(t)\Big|\boldsymbol{Q}_u(t),\boldsymbol{Q}_c(t)\Big\}. \tag{15}$$

where V is a non-negative tradeoff parameter that is chosen to adjust the cost and delay tradeoff, i.e., how much we care about the cost reduction compared to the processing delay. Then, our objectives for (NC-U), (NC-C) and (CP) are to minimize the objective functions (13), (14) and (15) in every time slot t, respectively. The key derivation step is to obtain upper bound on the Lyapunov drift-plus-penalty.

Deriving an upper bound. We derive upper bounds of (13)-(15) using queuing dynamics (2), (3) and bounds of computing job arrivals, CPU and network speeds assumed in Section II.

Lemma 1. *Under any possible control variables $\boldsymbol{\theta}_u(t)$, $\boldsymbol{s}_u(t)$, $\boldsymbol{\theta}_c(t)$ and $p(t)$, we have:*

$$\text{(NC-U): } \Delta L(t)\le B_1-\Big(\theta_c^i(t)\frac{s_c}{\gamma^i}-\theta_u^i(t)\mu^i(t)\Big)Q_c^i(t)-\Big((1-\theta_u^i(t))\frac{s_u^i(t)}{\gamma^i}+\theta_u^i(t)\mu^i(t)-A^i(t)\Big)Q_u^i(t)$$

$$\text{(NC-C): } \Delta L(t)\le B_2-\sum_{i\in\mathcal{I}}\Big(\theta_c^i(t)\frac{s_c}{\gamma^i}-\theta_u^i(t)\mu^i(t)\Big)Q_c^i(t)$$

$$\text{(CP): } \Delta L(t)\le B_3-\sum_{i\in\mathcal{I}}\Big(\theta_c^i(t)\frac{s_c}{\gamma^i}-\theta_u^i(t)\mu^i(t)\Big)Q_c^i(t)-\sum_{i\in\mathcal{I}}\Big((1-\theta_u^i(t))\frac{s_u^i(t)}{\gamma^i}+\theta_u^i(t)\mu^i(t)-A^i(t)\Big)Q_u^i(t)$$

$$-\sum_{i\in\mathcal{I}}\Big((1-\theta_u^i(t))\frac{s_u^i(t)}{\gamma^i}+\theta_c^i(t)\frac{s_c}{\gamma^i}-A^i(t)\Big)\big(Q_u^i(t)+Q_c^i(t)\big)$$

where

$B_1=\frac{1}{2}\Big((A_{\max}^i)^2+\frac{(s_{u,\max})^2}{(\gamma^i)^2}+2(\mu_{\max}^i)^2+\frac{(s_c)^2}{(\gamma^i)^2}\Big)$,

$B_2=\frac{1}{2}\sum_{i\in\mathcal{I}}\Big(\frac{(s_c)^2}{(\gamma^i)^2}+(\mu_{\max}^i)^2\Big)$,

$B_3=\sum_{i\in\mathcal{I}}\Big((A_{\max}^i)^2+\frac{(s_{u,\max})^2}{(\gamma^i)^2}+(\mu_{\max}^i)^2+\frac{(s_c)^2}{(\gamma^i)^2}+\frac{s_{u,\max}s_c}{(\gamma^i)^2}\Big)$.

Proof. The proof is presented in the technical report [16]. □

We develop NC-UC, NC-CC and CP-JC algorithms by finding (NC-U): $(\theta_u^i(t),s_u^i(t))$, (NC-C): $(\boldsymbol{\theta}_c(t),p(t))$ and

(CP): $(\boldsymbol{\theta}_u(t), \boldsymbol{s}_u(t), \boldsymbol{\theta}_c(t))$ which minimize the upper bounds of (13)-(15) every time slot.

C. Control algorithms for cost minimization of users and CSP in mobile cloud systems

First, we describe NC-UC and NC-CC algorithms for non-cooperation case as follows.

Algorithms for non-cooperation case: NC-UC, NC-CC

At each time slot t,

Cloud-side control, NC-CC:

1: Find a user set $\mathcal{J}$ such that $Vb(t) < \frac{s_c}{\gamma^j} Q_c^j(t), \forall j \in \mathcal{J}$.
2: Schedule $\boldsymbol{\theta}_c(t)^*$ by selecting top $\min(n(t), \|\mathcal{J}\|)$ number of user queues where the reference is $\frac{Q_c^j(t)}{\gamma^j}$.
3: Select $p(t)^* = \arg\min_{p(t)} \sum_{i\in\mathcal{I}} X^i(p(t))$, (16)
where $X^i(p(t)) = -\theta_u^i(t)\big(Vp(t)\gamma^i\mu^i(t) - \mu^i(t)Q_c^i(t)\big)$.

User-side control, NC-UC: for each user i,

4: **if** $\min_{s_u^i(t)\in\mathcal{S}_u} \big\{V\alpha^i(t)E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i}Q_u^i(t)\big\} \geq V\big(\alpha^i(t)E_N + p(t)\gamma^i\mu^i(t)\big) - \mu^i(t)\big(Q_u^i(t) - Q_c^i(t)\big)$, (17)
5: Select $\theta_u^i(t)^* = 1$ and $s_u^i(t)^* = 0$.
6: **else**
7: Select $\theta_u^i(t)^* = 0$ and $s_u^i(t)^* = \arg\min_{s_u^i(t)\in\mathcal{S}_u} \big\{V\alpha^i(t)E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i}Q_u^i(t)\big\}$.
8: **end**

Cloud-side algorithm for non-cooperation case (NC-CC): Cloud-side queue scheduling $\boldsymbol{\theta}_c(t)^*$ can be divided into two steps: 1) Find $\mathcal{J}$ which is the set of users whose cloud-side queue is preferred to be served. In other words, stabilizing cloud-side queue of user j is more important than the cost reduction for all $j \in \mathcal{J}$. 2) Schedule the maximum $n(t)$ number of queues in a descending order of $\frac{Q_c^j(t)}{\gamma^j}$ from the set $\mathcal{J}$ where $\frac{Q_c^j(t)}{\gamma^j}$ means the workload size in cycles. Step 1) means the CSP opportunistically stabilize cloud-side queue in terms of variation of electricity bills, and step 2) means the CSP fairly schedule cloud-side queues in terms of queueing delay among users. Next, the CSP selects cloud computing price $p(t)^*$ that minimizes (16) where $X^i(p(t))$ represents the tradeoff between the profit and queue stability of user i. In other words, the CSP regulates $p(t)$ to maximize cloud computing profit from users, and increases $p(t)$ to stabilize cloud-side queue by blocking excessive cloud computing requests from the users. However, we cannot separate problem (16) into each user and have to check all the $p(t) \in [0, \infty)$ to find optimal point which requires high complexity. We can find $p(t)^*$ easily without loss of optimality using Lemma. 2

Lemma 2. *We can find optimal $p(t)^*$ by checking L number of pricing points which is less than the number of users, I.*

Proof. For user i, (17) is a linear function of $p(t)$. Fig. 2 shows the two cases of $X^i(p(t))$.

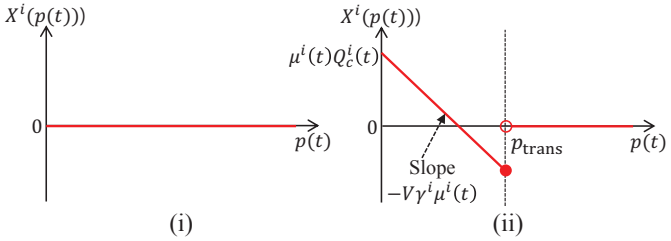


Fig. 2: Two cases of $X^i(p(t))$.

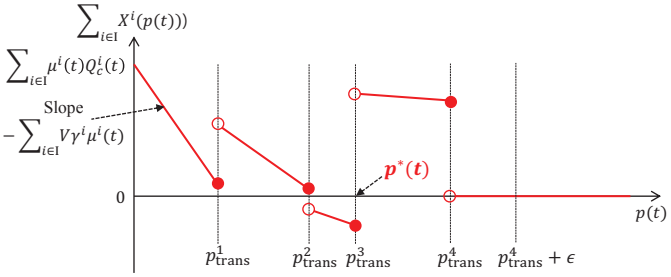


Fig. 3: Example of $\sum_{i\in\mathcal{I}} X^i(p(t))$ when $L = 4$.

(i) **If** $\min_{s_u^i(t)\in\mathcal{S}_u} \big\{V\alpha^i(t)E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i}Q_u^i(t)\big\} < V\alpha^i(t)E_N - \mu^i(t)\big(Q_u^i(t) - Q_c^i(t)\big)$,
Then, $\theta_u^i(t) = 0$ and $X^i(p(t)) = 0$ for all $p(t) \in [0, \infty)$.

(ii) **Else**, there exists a price p_{trans} called *transition price*, which makes

$$\theta_u^i(t) = \begin{cases} 1, \forall p(t) \in [0, p_{\text{trans}}], \\ 0, \forall p(t) \in (p_{\text{trans}}, \infty), \end{cases}$$

where $p_{\text{trans}} = \dfrac{\min_{s_u^i(t)\in\mathcal{S}_u} \big\{V\alpha^i(t)E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i}Q_u^i(t)\big\}}{V\gamma^i\mu^i(t)} + \dfrac{\mu^i(t)\big(Q_u^i(t) - Q_c^i(t)\big) - V\alpha^i(t)E_N}{V\gamma^i\mu^i(t)}$.

Then, $X^i(p(t)) = -Vp(t)\gamma^i\mu^i(t) + \mu^i(t)Q_c^i(t), \forall p(t) \in [0, p_{\text{trans}}]$ which is a linear decreasing function and $X^i(p(t)) = 0, \forall p(t) \in (p_{\text{trans}}, \infty)$ which is constant.

For the I number of users, there exist L number of *transition price* points where $L \leq I$. We index transition prices in ascending order. (i.e., $p_{\text{trans}}^1 \leq p_{\text{trans}}^2 \leq \ldots \leq p_{\text{trans}}^L$). For each interval $(p_{\text{trans}}^i, p_{\text{trans}}^{i+1}]$ where $i \in \{1, 2, \ldots, L-1\}$ it is enough to check p_{trans}^{i+1} to find minimum $\sum_{i\in\mathcal{I}} X^i(p(t))$ value because $\sum_{i\in\mathcal{I}} X^i(p(t))$ is a superposition of linear and non-increasing function. Therefore, we have to check only L ($p_{\text{trans}}^i, i \in \{2, \ldots, L\}$ and $p_{\text{trans}}^L + \epsilon$) number of price points to find $p(t)^*$. Fig. 3 shows the example of how to find $p(t)^*$ for multi-user scenario when $L = 4$. □

User-side algorithm for non-cooperation case (NC-UC): Each user considers queue lengths of user-side and cloud-side, network states, and cloud computing price at time slot t. The user decides $\theta_u^i(t)$ by comparing (13) which represents tradeoff between the user's cost and queue stability of both user-side and cloud-side. We can see that the user tries to schedule $\theta_u^i(t) = 1$ (processing by cloud) when the user-side queue

lengths are much larger than cloud-side one ($Q_u^i(t) \gg Q_c^i(t)$), network channel states are good and cloud computing price is low. This implies the user distributes the computing job opportunistically to the user-side and cloud-side. If $\theta_u^i(t) = 0$, the user increases CPU clock speed to stabilize the user-side queue when the user-side queue lengths become larger.

Second, we describe a CP-JC algorithm for cooperation case as follows.

Algorithm for cooperation case: CP-JC

At each time slot t,

Cloud-side control:

1: Find a user set $\mathcal{J}$ such that $Vb(t) < \frac{s_c}{\gamma^j} Q_1^i(t), \forall j \in \mathcal{J}$.
2: Schedule $\boldsymbol{\theta}_c(t)^*$ by selecting top $\min(n(t), \|\mathcal{J}\|)$ number of user queues where the reference is $\frac{Q_1^i(t)}{\gamma^j}$,

User-side control: for each user i,

3: **if** $\min_{s_u^i(t) \in \mathcal{S}_u} \{V\alpha^i(t) E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i}(Q_2^i(t))\} \geq V\alpha^i(t)E_N - \mu^i(t)\big(Q_u^i(t) - Q_c^i(t)\big)$, (18)
4: Select $\theta_u^i(t)^* = 1$ and $s_u^i(t)^* = 0$.
5: **else**
6: Select $\theta_u^i(t)^* = 0$ and $s_u^i(t)^* = \arg\min_{s_u^i(t) \in \mathcal{S}_u} \{V\alpha^i(t) E_C(s_u^i(t)) - \frac{s_u^i(t)}{\gamma^i} Q_2^i(t)\}$.
7: **end**

where $Q_1^i(t) = Q_u^i(t) + 2Q_c^i(t)$ and $Q_2^i(t) = 2Q_u^i(t) + Q_c^i(t)$

Algorithm for cooperation case (CP-JC): A CSP controls $\boldsymbol{\theta}_c(t)^*$ by similar mechanism with NC-CC. However, the CSP considers both user and cloud-side queues, while NC-CC considers only cloud-side one. It implies that the CSP also cares queue stability of user-side to optimize social objective for the cooperation case. All mobile users schedule $\theta_u^i(t)^*$ by similar mechanism with NC-UC, but the computing price term $p(t)\gamma^i\mu^i(t)$ is disappeared (in (18)) that means the user decides offloading policy with considering only the energy cost and queue stability. The reason for theses phenomenon is that the user behaves to minimize the social cost.

IV. Trace-Driven Simulation

In this section, we evaluate the proposed algorithms through real measurement and trace driven simulations.

A. Real Measurement and Traces

Real energy measurement. We measure the energy consumptions of Galaxy Note smartphone [19] with LTE chipset using Monsoon power monitor [20]. We measure the energy consumptions of CPU for several clock speeds (0.1∼1.4GHz) and fit the parameters (κ, φ, ρ) to the typical CPU energy consumption model in [6] as follows where Δt denotes time duration for one time slot in sec[4].

$$E_C(s_u^i(t)) = (\kappa(s_u^i(t))^\varphi + \rho)\Delta t \quad (19)$$

[4]All measurements are done with disabling other components, GPS, display (and network in CPU measurement).

The measured network energy consumption (LTE) and CPU energy parameters are 2605mJ/sec, 0.33 (κ), 3.0 (φ) and 0.10 (ρ), respectively.

Real traces. We collect the eight numbers of LTE uplink throughput traces for three network service providers and the diverse mobility scenarios including fixed location, driving in downtown, driving in highway and KTX (Korea Train eXpress). The LTE throughput traces are collected by uploading 5MB size of dummy files to our private server and checking transmission time every 1 minute within active time zones (11:00AM to 10:00 PM). The measured average uplink throughput is 6.4Mbps. To run the simulation, we use the uplink throughput traces for the users by randomly selecting the throughput traces. To generate computing job arrivals of users, we use a YouTube video size distribution dataset in [21] and re-scale it, where the average file size is 12.6MB. We use the real-world traces of electricity bills in California, USA [8] where the time granularity is 5 minutes.

B. Simulation Setup

We consider a scenario that mobile users who have Galaxy Note smartphone move around in the LTE coverage. They run cloud offloadable applications where the jobs arrive as a Bernoulli process. For simplicity, the processing densities of all users are set to be the same from 200 to 4000 cycles/bit. We mainly run simulations for the 2000cycles/bit of processing density which is generally acceptable for video transcoding, chess game and face recognition applications [9], [6]. We fix the energy-price tradeoff parameter of all users as $\alpha^i(t) = 2.44 \times 10^{-5}$\$/Joule for easy analysis. On the cloud-side, we assume that the CSP has cloud servers where the available number of them is time-varying between 1 to the number of users. The electricity bill is also time-varying [8] where the average is 5.5×10^{-5}\$ for activating one server. Detailed simulation parameters are summarized in Table II.

TABLE II: Simulation settings

Number of users	80
Experimental time [hour]	6
Avg. arrival rate [Mbps]	0.96
Avg. electricity price [\$/server]	5.5×10^{-5}
Available # of servers at CSP	$uniform[1, 80]$
Capacity of one server [GHz]	4.0
Energy-price tradeoff parameter [\$/Joule]	2.44×10^{-5}

As performance metrics, we consider average costs and queue lengths for the users and CSP. By little's law [22], the average delay can be interpreted as the average queue lengths divided by the average arrival rate. We compare our algorithms, NC-UC and NC-CC under non-cooperation case with existing cloud offloading algorithms at user-side [9]–[11], and cloud-side with flat-pricing and scheduling cloud-side queue without delay. For all comparing algorithms, CPU speed is selected by a DVFS scheme [23]. For MAUI [9], cloud offloading is selected only when the current workloads can be served within a specified delay. For ThinkAir [10],

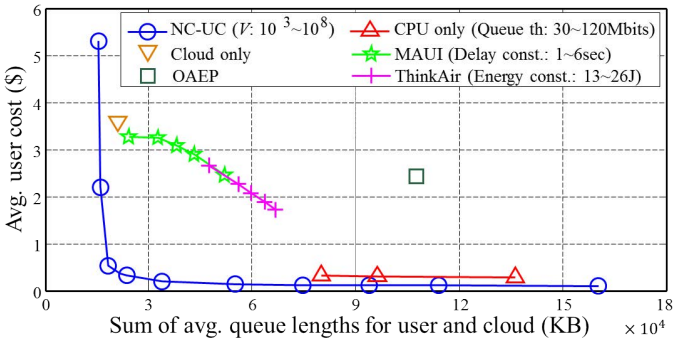


Fig. 4: Avg. user cost and sum of avg. queue lengths for user and cloud for NC-UC and existing user-side control algorithms.

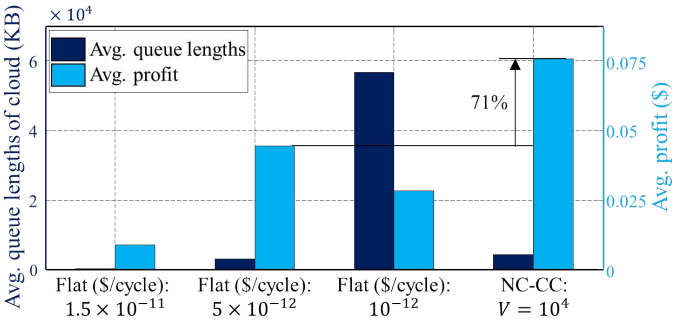


Fig. 5: Avg. queue lengths of cloud and avg. profit of CSP for NC-CC and flat pricing algorithms.

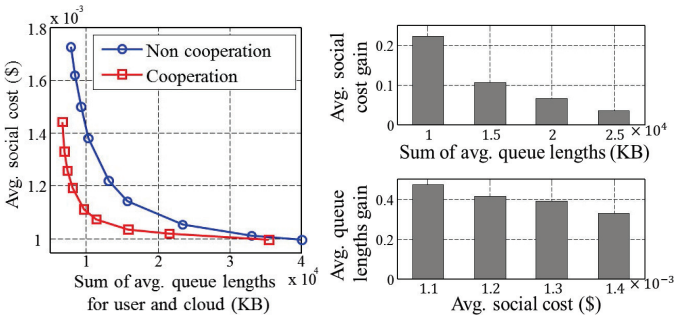
cloud offloading is selected only when both delay and network energy are less than constraints. For OAEP[5] [11], could offloading is selected when the energy efficiency (speed/energy) of CPU is greater than network resource. CPU only and Cloud only policies are always using the local computing and cloud computing, respectively. We also compare our CP-JC algorithm with (NC-UC+NC-CC) to verify the advantage of cooperation between the users and CSP.

C. Simulation Results 1 - Non-cooperation Case

Cost-delay tradeoff. Fig. 4 depicts the average cost of users and the sum of average queue lengths stabilized for user-side and cloud-side for NC-UC (when the CSP adopts NC-CC) and existing algorithms. First, we observe the 83% cost savings for NC-UC by trading only 0.6MB queue lengths. The cost saving is due to that *(i)* it saves the CPU energy cost by adjusting local CPU clock speed and *(ii)* it opportunistically transfers computing job to the cloud by taking into account network states, the price for MCC service and remaining jobs.

Second, we observe that the NC-UC algorithm outperforms existing algorithms where the cost savings are 63% (8.0MB queue lengths) for CPU only, 89% (2.1MB queue lengths) for Cloud only, 91% (2.4MB queue lengths) for MAUI, 94% (4.7MB queue lengths) for ThinkAir, and 96% (10.7MB queue lengths) for OAEP. The reason why NC-UC outperforms other algorithms is that it jointly optimizes cloud offloading and CPU speed selection by considering the time varying network states, cloud computing prices and remaining jobs.

[5]We denote by OAEP the optimal application execution policy in [11]



(a) Avg. social cost and sum of avg. queue lengths tradeoff curves. (b) Avg social cost and queue lengths gain.

Fig. 6: Avg. social cost and sum of avg. queue lengths for user and cloud for the non-cooperation (NC-UC+NC-CC) and the cooperation (CP-JC).

Impact of dynamic pricing and exploiting opportunity of time-varying electricity bill. We compare flat-pricing algorithms to our NC-CC when users adopt NC-UC. Fig. 5 demonstrates the average queue lengths and the profit (gained money paid by users minus electricity bill) of the CSP. For the flat pricing, as the computing price becomes cheaper, the cloud-side queue lengths become larger due to the increasing requests of cloud computing from the users. However, the average profit of the CSP is not monotonic to the computing price and the highest profit achievable price is $5{\times}10^{-12}$ \$/cycle between $1.5{\times}10^{-11}$ \$/cycle and 10^{-12} \$/cycle. The NC-CC algorithm achieves 71% of the profit gain compared to the flat-pricing (5×10^{-12} \$/cycle) while maintaining similar queue lengths. It is because NC-CC fully exploits the degree of willingness to pay of users by time-dependent pricing, and grabs opportunities time-varying electricity bills which can save the operating cost of cloud servers with satisfying the queue stability.

D. Simulation Results 2 - Cooperation Gains

Cost-delay tradeoff. We compare two proposed dual-side control algorithms when the users and CSP cooperate (CP-JC) or not (NC-UC+NC-CC). Fig. 6 demonstrates the social cost (for all users and CSP) and the sum of average queue lengths stabilized for user-side and cloud-side. Fig. 6(a) reveals that the social cost for cooperation case (CP-JC) is lower than that of non-cooperation case (NC-UC+NC-CC) for all average queue lengths. In Fig. 6(b), the cooperation achieves 22% of cost gain when the average queue lengths are 10^4KB and 47% of delay gain when the average cost is $1.1\times10^{-}3$\$, compared to the non-cooperation case. As the queue lengths (the cost) become larger, the cost gain (queue length gain) becomes smaller because both of control algorithms similarly operate when the cost-delay tradeoff parameter V is extremely large or small.

Impact of processing density. Fig. 7 demonstrates the ratio of computing job processed by the CSP over total volume of processed jobs as a function of processing density, γ^i, from 200 to 4000cycles/bit. We could find two interesting results: *(i) The ratio of jobs processed by the CSP for the cooperation case is higher than that for the non-cooperation case.* It is

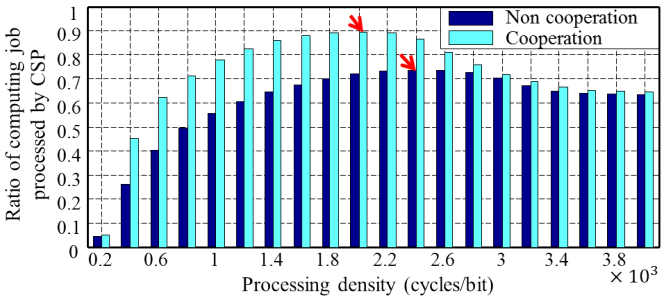


Fig. 7: Ratio of computing job processed by CSP for the different processing densities (200∼4000cycles/bit) when $V = 6 \times 10^{12}$.

because the users transfer the jobs to the CSP without burden of computing price and the CSP schedules cloud-side queues with considering the remaining jobs at the user-side when the users and CSP cooperate with each other. These phenomenas make the social cost become smaller which are demonstrated in the simulation results in Fig. 6.

(ii) As the processing density increases, the ratio increases to a certain point of processing density marked by red arrow in Fig. 7 and decreases again from that point. The reasons are as follows: 1) As the processing density goes higher to the marked point, it requires faster CPU clock speed which consumes higher CPU energy than networking energy; hence the cloud computing is likely to be more preferred than local computing. 2) As the processing density goes extremely higher[6] over the marked point, the cloud servers may become bottleneck because the cloud servers have finite processing capacities. Then, the users prefer local computing rather than cloud offloading; hence the ratio processed by the CSP decreases again. These interesting results is due to that we concretely model both of local CPU DVFS capability of mobile users and finite processing capacity of cloud servers, which were not entirely captured in the previous cloud offloading studies, e.g., [11]–[14].

V. Concluding Remark

In this paper, we explored the cost-delay tradeoffs of mobile users and cloud service provider (CSP) in mobile cloud computing systems. We proposed two kinds of dual-side control algorithms for cost minimization which is composed of energy and pricing cost, for given delay constraints. This paper is first to study non-cooperation and cooperation scenarios between the users and CSP, and propose user-side and cloud-side control algorithms for the non-cooperation and dual-side control algorithm for the cooperation. Each mobile user controls resource scheduling of computing jobs and local CPU clock frequency, and the CSP controls cloud-side queue scheduling and cloud computing price for using mobile cloud computing service. Trace-driven simulations based on real measurement demonstrate that our algorithms for non-cooperation and cooperation scenarios achieve drastic cost saving for users and CSP, respectively; moreover, the results provide us a message that realistic modeling for both of mobile users, e.g., DVFS capability and the CSP, e.g., finite processing capacity of cloud servers is essential for the analysis of the mobile cloud computing.

[6]Note that there exist computing jobs with extremely high processing density such as virus scanning (32946∼36992cycles/bit) [24] and N-queens puzzle (87.8∼8250cycles/bit) [10].

References

[1] M. Latouche, C. Rauschen, O. Oszabo, J. Creusat, and W. Belmans, "Mobile data explosion: How mobile service providers can monetize the growth in mobile data through value-added services," 2013. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/sp/Monetizing-the-Mobile-Data-Explosion.pdf
[2] "Windows Azure." [Online]. Available: http://azure.microsoft.com/
[3] "Amazon EC2." [Online]. Available: http://aws.amazon.com/ec2/
[4] "Google Cloud Flatform." [Online]. Available: https://cloud.google.com/solutions/mobile/
[5] "Mobile cloud applications." [Online]. Available: http://abiresearch.com/research/1003385
[6] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Dynamic speed scaling for energy minimization in delay-tolerant smartphone applications," in *Proc. of IEEE INFOCOM*, Toronto, Canada, Apr. 2014, pp. 2292–2300.
[7] M. Jia, J. Cao, and L. Yang, "Heuristic offloading of concurrent tasks for computation-intensive applications in mobile cloud computing," in *Proc. of IEEE INFOCOM Workshop on Mobile Cloud Computing*, Toronto, Canada, Apr. 2014, pp. 358–362.
[8] "California iso." [Online]. Available: http://www.caiso.com/
[9] E. Cuervo, A. Balasubramanian, and D. Cho, "MAUI: Making smartphones last longer with code offload," in *Proc. of ACM MobiSys*, San Francisco, CA, USA, June 2010, pp. 49–62.
[10] 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. of IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 945–953.
[11] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and D. Wu, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. on Wireless Communications*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[12] M. Lin, A. Wierman, L. Andrew, and E. Thereska, "Dynamic right-sizing for power-proportional data centers," *IEEE/ACM Trans. on Networking*, vol. 21, no. 5, pp. 1378–1391, Oct. 2013.
[13] Q. Wang, K. Ren, and X. Meng, "When cloud meets ebay: towards effective pricing for cloud computing," in *Proc. of IEEE INFOCOM*, Orlando, FL, USA, Mar. 2012, pp. 936–944.
[14] S. Ren and M. van der Schaar, "Dynamic scheduling and pricing in wireless cloud computing," *IEEE Trans. on Mobile Computing*, vol. 13, no. 10, pp. 2283–2292, Oct. 2014.
[15] I. Menache, A. Ozdaglar, and N. Shimkin, "Socially optimal pricing of cloud computing resources," in *Proc. of ACM VALUETOOLS*, ICST, Brussels, Belgium, 2011, pp. 322–331.
[16] Y. Kim, J. Kwak, and S. Chong, "Dual-side dynamic controls for cost minimization in mobile cloud computing systems," *Technical Report*, Jan. 2015. [Online]. Available: http://netsys.kaist.ac.kr/publication/papers/Resources/dual_side.pdf
[17] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, pp. 1–211, 2010.
[18] M. Ra, J. Peak, A. Sharma, R. Govindan, M. Krieger, and M. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. of ACM MobiSys*, SF, California, USA, June 2010, pp. 255–270.
[19] "Samsung Galaxy Note specifications." [Online]. Available: http://www.samsung.com/global/microsite/galaxynote/note/spec.html?type=find
[20] "Monsoon Power Monitor." [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/
[21] "Dataset for statistics and social network of YouTube videos." [Online]. Available: http://netsg.cs.sfu.ca/youtubedata/
[22] L. Kleinrock, *Queuing systems*. Wiley, 1975.
[23] "Kernel governors, modules, I/O scheduler, CPU tweaks AIO app configs." [Online]. Available: http://forum.xda-developers.com/showthread.php?t=1369817
[24] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Proc. of ACM EuroSys*, Salzburg, Austria, Apr 2011, pp. 301–314.