Processor-Network Speed Scaling for Energy–Delay Tradeoff in Smartphone Applications

Jeongho Kwak, Okyoung Choi, Song Chong, *Member, IEEE*, and Prasant Mohapatra, *Fellow, IEEE*

***Abstract*—Many smartphone applications, e.g., file backup, are intrinsically delay-tolerant so that data processing and transfer can be delayed to reduce smartphone battery usage. In the literature, these energy–delay tradeoff issues have been addressed independently in the forms of Dynamic Voltage and Frequency Scaling (DVFS) problems and network selection problems when smartphones have multiple wireless interfaces. In this paper, we jointly optimize the CPU speed and network speed to determine how much more energy can be saved through the joint optimization when applications can tolerate delays. We propose a dynamic speed scaling scheme called SpeedControl that jointly adjusts the processing and networking speeds using four controls: application scheduling, CPU speed control, wireless interface selection, and transmit power control. Through invoking the "Lyapunov drift-plus-penalty" technique, the scheme is demonstrated to be near optimal because it substantially reduces energy consumption for a given delay constraint. This paper is the first to reveal the energy–delay tradeoff relationship from a holistic perspective for smartphones with multiple wireless interfaces, DVFS, and multitasking capabilities. The trace-driven simulations based on real measurements of CPU power, network power, WiFi/3G throughput, and CPU workload demonstrate that SpeedControl can reduce battery usage by more than 42% through trading a 10 minutes delay when compared with the same delay in existing schemes; moreover, this energy conservation level increases as the WiFi coverage extends.**

***Index Terms*—CPU speed scaling, energy–delay tradeoff, heterogeneous wireless networks, multitasking, network interface selection, transmit power control.**

I. Introduction

SMARTPHONES now comprise a large part of our lives. Recent surveys have reported that daily usage time of smartphones is between 3.5 and 5 hours [2], [3], and 89% of smartphone users use their smartphones throughout the day [4]. Furthermore, many current applications are generating significant amounts of mobile traffic. Cisco forecasts that a single smartphone could generate as much network traffic as 37 feature phones in 2014; moreover, smartphones will reach three-quarters of mobile data traffic by 2019 [5]. For these reasons, the energy consumption of smartphones is drastically increasing. In particular, the energy consumption for processing and transferring data in smartphone applications is increasing.

The maximum CPU clock frequency is continually increasing (e.g., the latest Qualcomm mobile chipset has a maximum of 2.5 GHz CPU clock frequency [6]) to meet the increasing demands of applications and users. Operating at the maximum CPU clock frequency results in significant amount of energy consumption as the CPU power consumption is a superlinearly increasing function of the clock frequency. Smarphones have multiple network interfaces including cellular and WiFi networks, which tend to facilitate more networking applications with higher data rates that yield higher networking energy consumption.

In the CPU context, many application processors (APs) for smartphones support Dynamic Voltage and Frequency Scaling (DVFS) [6]–[8], which controls the CPU clock frequency and voltage depending on the CPU workload. The DVFS exploits power saving opportunities because the CPU power consumption depends superlinearly on the CPU clock frequency; however, this power saving comes at the cost of increasing processing delays. For example, our measurements in Section V-A demonstrate that lowering the CPU speed of a Galaxy Note smartphone from 1.4 GHz to 0.7 GHz reduces 80% of the CPU power while the processing delay is twofold.

In the network context, the selection between multiple network interfaces and the transmit power control (which results in data rate changes) exploits the power saving opportunities as a result of the following two reasons. *(i)* The power consumption for transmitting a single bit differs in each network interface. Our measurements in Section V-A verify that the WiFi network interface of the Nexus S smartphone is 5.5 times more energy efficient than that of the 3G network for transmitting the same quantity of data. *(ii)* The power consumption of network interfaces for transmitting a single bit can be reduced when the transmit power is reduced due to the transmit power being an exponentially increasing function of the data rate from the transformation of the ideal Shannon capacity [9]. To the best of our knowledge, studies on the joint optimizations of CPU speed scaling (i.e., DVFS policy) and network speed scaling (i.e., network selection and transmit power control policies) have not been undertaken for energy minimization.

Manuscript received April 17, 2014; revised October 17, 2014, February 02, 2015, and March 04, 2015; accepted March 30, 2015; approved by IEEE/ACM TRANSACTIONS ON NETWORKING Editor M. Meo. Date of publication April 21, 2015; date of current version June 14, 2016. This work was supported in part by the ICT R&D program of MSIP/IITP [14-000-04-001, Development of 5G Mobile Communication Technologies for Hyper-connected smart services], and in part 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), and in part by the National Science Foundationthrough the grant CNS-1319721. Part of this work was presented at the IEEE INFOCOM 2014, Toronto, ON, Canada.

J. Kwak, O. Choi, and S. Chong are with the Department of Electrical Engineering, Korea Advanced Institute of Science and Technology (KAIST), Daejeon 305-701, Korea (e-mail:jh.kwak@netsys.kaist.ac.kr; okyoung@netsys.kaist.ac.kr; songchong@kaist.edu).

P. Mohapatra is with the Department of Computer Science, University of California, Davis, CA 95616 USA (e-mail: pmohapatra@ucdavis.edu).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TNET.2015.2419219

Many smartphone applications are likely to use both processing and networking resources as well as having delay-tolerant natures.[1] For these applications, the controls in the CPU and network will affect each other; thus, independent controls for the CPU speed, network selection, and transmit power result in energy inefficiencies. For example, consider a situation where a smartphone runs an application that processes a file (e.g., transcoding a video file) and then uploads the transcoded file to a cloud server, e.g., Dropbox [11]. Assume that the network condition is bad, e.g., only a 3G link with a low data rate is available. As the CPU workload increases in this situation, DVFS would continue to increase the data processing speed accordingly resulting in the network queue becoming a bottleneck with a large backlog because the data processing speed would exceed the low data transmission speed at some point. This situation should be avoided from an energy efficiency perspective because the CPU would operate at an unnecessarily fast speed and consequently waste energy.

In contrast, if the backlog in the network queue was small, the data transmission could be postponed until the device locates an energy efficient network such as WiFi, or selects a low transmit power in a WiFi network. Note that the average energy consumption in transmitting a single bit over WiFi networks is significantly less than that required in 3G networks (see the measurements in Section V-A). Therefore, the situation described above is also harmful in terms of energy efficiency because the network queue would transmit data over the energy inefficient links or increase the transmit power (which results in a high data rate) due to the large backlog and it cannot wait for an energy efficient network to be located.

According to our measurements in Section V-A, the CPU power consumption required to process one bit for typical transcoding applications is comparable with the power consumption of WiFi or 3G interfaces to transmit one bit. Thus, power management for the CPU and network interfaces are equally important, and this supports the argument that the CPU speed and network speed must be jointly optimized.

In a recent survey of the Top 50 smartphone applications in Google Play [12], 44% of the top applications were networking applications (NAs) and 56% were non-networking applications (NNAs). Contemporary smartphone operating systems provide multitasking capabilities (e.g., iOS and Android [13], [14]); therefore, many smartphones run the two kinds of applications simultaneously. NAs use both CPU and network resources, whereas NNAs use CPU resources only. If the two kinds of applications share the CPU queue, they interfere with each other. Therefore, a method of isolating the performance of NNAs from that of NAs should be considered in the joint optimization of the CPU and network speed scaling.

In this paper, we propose a dynamic speed scaling scheme called SpeedControl that simultaneously adjusts both the processing speed and networking speed using four controls: application scheduling, CPU speed control, network (wireless interface) selection, and transmit power control. The contributions of the paper are summarized as follows.

[1]These types of applications do not need to instantaneously transmit or process the data, i.e., delaying the data transmission or processing does not be critical for user satisfaction in the delay-tolerant applications [10].

- The proposed SpeedControl algorithm is the first to simultaneously optimize CPU speed scaling and network speed scaling, which incorporates the network interface selection and transmit power control, in order to minimize energy consumption in delay-tolerant smartphone applications. It is demonstrated that the proposed algorithm is near optimal because it substantially reduces the energy consumption for a given delay constraint.
- SpeedControl is the first algorithm to address the co-existence issues of networking/non-networking applications sharing the CPU resource in a smartphone with multitasking capabilities. To isolate the performance of non-networking applications from that of networking applications in this highly coupled environment, an application scheduling policy that determines the sequence of CPU access between the two kinds of applications is incorporated in the SpeedControl algorithm.
- SpeedControl not only significantly outperforms existing schemes but also does not affect the performance of background non-networking applications. Trace-driven simulations based on real measurements of operational environments and system parameters demonstrate that SpeedControl can reduce smartphone battery usage by more than 42% by trading approximately 10 minutes of transfer delay when compared with the existing schemes when the WiFi temporal coverage is 65%. Moreover, the energy saving tendency increases as the WiFi temporal coverage increases. We also verify the practicality of the simulation results through prototype experiments using an Android smartphone with SpeedControl installed.

The remainder of the paper is organized as follows: we begin with discussing the related work in Section II. In Section III, we describe the proposed system model. Then, we describe the problem formulation and derive the joint application scheduling, CPU speed control, network interface selection, and transmit power control algorithm, named SpeedControl, and demonstrate theoretical analysis in Section IV. Next, in Section V, we evaluate the proposed SpeedControl algorithm through measurements, trace-driven simulations and experiments. Finally, we conclude the paper in Section VI.

II. Related Work

DVFS. There have been extensive studies on the energy–delay tradeoff in network devices such as cellular base station or router using DVFS [15]–[17]. Son *et al.* [15] suggested an energy efficient joint control algorithm of DVFS and user association using the fact that the power consumption of base station is well modeled by a cubic polynomial scaling of the processing speed. Wierman *et al.* [16] analyzed the optimal energy–delay tradeoff under several processing power models including static and dynamic speed scaling.

Several DVFS techniques have been considered in mobile devices [6]–[8]. Liang *et al.* [8] demonstrated that there exist a critical CPU clock speed to minimize the energy consumption of handheld devices. Recent mobile chipsets, e.g., Snapdragon S4 [6], have also adopted DVFS techniques for energy efficiency. However, their DVFS policies are far from optimal. For instance, in case of the Ondemand policy [18], the CPU speed is set to maximum when the amount of workloads exceeds a certain threshold, which is controlled manually, and then the smartphone gradually decreases the speed depending on the workloads.

Network selection. Recently, the interests on network selection for delay-tolerant applications considering the battery consumption of smartphones, which include [19]–[22], have been increasing. Rahmati *et al.* [19] suggested on-the-spot network selection by examining the tradeoff between energy consumption for WiFi search and transmission efficiency when the WiFi networks are intermittently available. Some studies suggested delayed network selection policies by exploring the tradeoff between the transmit power of heterogeneous network interfaces (3G and WiFi) and transmission delay [20], [21]. Lee *et al.* [22] demonstrated that 20% power saving can be achieved by permitting 1 hour delay for an application having strict delay constraints by exploiting the delayed WiFi offloading. The network selection also can indirectly affect the energy saving of cellular infrastructure by offloading the traffic to WiFi networks or exploiting wireless channel opportunism.

Transmit power control. High data rate can be achieved by using high transmit power for a given target bit error rate (BER) in general wireless networks. Recently, some works have suggested control schemes of transmit power and/or data rate (i.e., MCS level) with objectives of the energy minimization or throughput maximization under given delay or BER constraints in WiFi networks [23]–[25]. Especially, Xu *et al.* [24] suggested an energy-efficient data rate control algorithm using the network queue information. For example, they increase the transmit power when the average queue lengths increase so as to stabilize the network queue. Also, Ismail *et al.* [26], [27] suggested transmit power control algorithms of mobile devices in heterogeneous networks for energy minimization with target QoS (quality of service) of video when the mobile devices are able to simultaneously transmit the video file through multiple wireless networks (e.g., cellular and WiFi), called multi-homing.

General energy–delay tradeoff framework. Assume that the objective of the network devices is to minimize the processing or networking power consumption while the total backlog of the devices is stabilized. Many energy minimization problems in the context of the data networks fall into this general framework. Neely [28] suggested a general solution form of this framework which called "Lyapunov drift-plus-penalty". For instance, the delayed network selection works [20], [21] used the above solution form in terms of controlling the networking speed to minimize the network interface power of a smartphone for given delay constraints. Also, Yao *et al.* [29] explored a trade-off between the networking energy and average transferring delay to minimize the power consumption of data centers by controlling the network routing and the number of active backend servers. Unfortunately, this framework intrinsically cannot guarantee the strict delay constraints. Therefore, there exist several studies trying to take account of the strict or instantaneous delay constraints by dynamically controlling the energy–delay tradeoff parameter or introducing delay sensitivity functions for different delay features [20], [30].

III. System Models

A. Application and Traffic Arrival Models

We consider two types of delay-tolerant applications. One is networking application (NA) which uses both processing and networking resources and the other is non-networking application (NNA) which uses only processing resources in a smartphone. For example, a smartphone user records two video clips using a camera application. The one of recorded video clips is transcoded and uploaded to a cloud server, e.g., Dropbox [11] (NA), and the other video clip is just transcoded in the smartphone (NNA). Let a set of applications be $\mathcal{K} = \{1, \ldots, K\}$. For simplicity, we consider only one NA and the other NNA. However, it can be easily generalized to more applications. We consider a time-slotted system indexed by $t = \{0, 1, \ldots\}$ where the interval is Δt. For each time slot t, workload arrival rates are $A_{\mathrm{NA}}(t)$ for NA and $A_{\mathrm{NNA}}(t)$ for NNA, respectively.

B. CPU and Network Models

We assume that a smartphone has one CPU core which handles several applications running in the smartphone. Workloads of each application demand different CPU processing resources. We call this notion as *processing density* (in cycles/bit) γ_{NA} for NA, γ_{NNA} for NNA which are defined as the average number of CPU cycles required per bit when the application is processed by the CPU [31]. Smartphones with DVFS capability adjust CPU speed $s(t) \in \{s_1, s_2, \ldots s_{max}\}$ (in cycles/Δt) [2] every time slot t depending on the system policy [6].

We assume that a smartphone is able to use two wireless network interfaces (i.e., cellular and WiFi) for data transfer. It is possible to select wireless interfaces such as 3G[3] or WiFi for data transfer in typical smartphones [13], [14]. However, the data transmission via cellular or WiFi networks is likely to be intermittently viable due to the mobility of smartphone users. We denote the time varying network availability of the smartphone at time slot t by $B(t) \in \{\{C, W, N\}, \{C, N\}, \{W, N\}, \{N\}\}$ where C is cellular, W is WiFi and N is none of them. Note that the smartphone is not able to simultaneously use the cellular and WiFi networks, i.e., it does not support multi-homing technology. Then, the smartphone is able to select a wireless interface $l(t)$ between cellular (C) and WiFi (W) or may not select any interface (N) every slot t, i.e., $l(t) \in B(t)$, depending on the system condition and network availability.[4]

We also consider a selection of data rates corresponding to Modulation and Coding Scheme (MCS) levels for the network interface $l(t)$.[5] For a certain target bit error rate (BER), different data rates require different signal-to-noise ratios (SNRs) [23], which are determined by transmit powers and wireless channel gain at slot t. Because the wireless channel is varying every slot,

[2]A set of available CPU speeds is determined by the Kernel in the operating system of the smartphone.

[3]It can be applied to 4 G or LTE technology without any modification of our system model.

[4]Time scale issue of the time slot t will be addressed in Section V-B.

[5]Even though the current smartphone OSs [13], [14] do not support Application Programming Interface (API) of the data rate selection, we think it is a natural extension of the current network controls at the API levels.

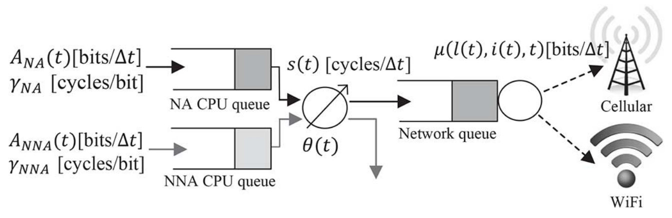

$A_{NA}(t)$[bits/Δt]
γ_{NA} [cycles/bit]
NA CPU queue
$A_{NNA}(t)$[bits/Δt]
γ_{NNA} [cycles/bit]
NNA CPU queue
$s(t)$ [cycles/Δt]
$\theta(t)$
$\mu(l(t), i(t), t)$[bits/Δt]
Network queue
Cellular
WiFi

Fig. 1. Queueing model for CPU and network parts in a smartphone.

let a feasible transmission mode (i.e., a combination of data rate and transmit power for a fixed BER target) be $\mathcal{I}_{l(t)}(t)$ every slot t for network interface $l(t)$. Then, we select one transmission mode index $i(t) \in \mathcal{I}_{l(t)}(t)$ every slot t. Because selecting the transmit power results in selecting the data rate or MCS level, we henceforth call this just as transmit power control. Once the smartphone selects a network interface $l(t)$ and a transmit power index $i(t)$ at slot t, the achievable uplink throughput is denoted by $\mu(l(t), i(t), t)$.

C. Queueing Model

We consider a queueing model as illustrated in Fig. 1. The queueing model is designed to isolate the performance of NNA from that of NA. NNA should not be affected by network environments because it does not use networking resources. However, if we design a queueing model so that packets for the two kinds of applications are not segregated for processing, the system cannot isolate the performance of NNA from that of NA when the network queue becomes a bottleneck, and NNA workloads are located in the behind of NA workloads at the CPU queue. In this situation, the NNA workloads cannot be scheduled even though the NNA does not use the networking resources. This is the reason why workloads for the two kinds of applications are distinguished and processed separately in our queueing model.

For the queueing model, $Q^c(t)$ denotes total CPU queue lengths at slot t, $Q^c_{\mathrm{NA}}(t)$ and $Q^c_{\mathrm{NNA}}(t)$ denote CPU queue lengths for the NA and NNA at slot t, respectively, i.e., $Q^c(t) = Q^c_{\mathrm{NA}}(t) + Q^c_{\mathrm{NNA}}(t)$, and $Q^n(t)$ denotes network queue lengths at slot t. An application scheduling indicator of the CPU processor at slot t is denoted by $\theta(t) \in \{0, 1\}$, e.g., if $\theta(t)$ is 0, the NNA is scheduled. We assume that a unit of the queue lengths is a bit, thus the CPU speed $s(t)$ (in cycles/Δt) should be divided by the processing density (cycles/bit) γ_{NA} or γ_{NNA} in the queueing model for a unit agreement. Then, the queue lengths of all CPU and network queues are updated as follows:

$$Q^c_{\mathrm{NA}}(t+1) = \left[Q^c_{\mathrm{NA}}(t) - \frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}} + A_{\mathrm{NA}}(t)\right]^+ \text{[bits]} \quad (1)$$

$$Q^c_{\mathrm{NNA}}(t+1) = \left[Q^c_{\mathrm{NNA}}(t) - \frac{(1-\theta(t))s(t)}{\gamma_{\mathrm{NNA}}} + A_{\mathrm{NNA}}(t)\right]^+ \text{[bits]} \quad (2)$$

$$Q^n(t+1) = \Bigg[Q^n(t) - \mu(l(t), i(t), t) + \theta(t)\min\left\{Q^c_{\mathrm{NA}}(t), \frac{s(t)}{\gamma_{\mathrm{NA}}}\right\}\Bigg]^+ \text{[bits]}. \quad (3)$$

D. Power Model

In the domain of processor design, the CPU power model has typically two different types which are introduced in [16]: static speed scaling and gated-static speed scaling. Because our real power measurement study in Section V-B reveals that contemporary smartphones consume some static power when the processor is not busy, we assume the CPU power consumption of smartphones is modeled by the static speed scaling as follows:

$$P^c(s(t)) = \alpha s(t)^x + \beta \quad (4)$$

where α, β and x are constants associated with the smartphones, and x is known as between 1–3 in typical network devices [17].

Usual smartphones have two power consuming parts in the network interfaces: One is transmit power and the other is idle power for both cellular and WiFi interfaces.[6] Because two network interfaces use different association methods and transmit powers to transfer data [32], the network power consumption $P^n(l(t), i(t), t)$ depends on the selected network interface $l(t)$ and transmit power index $i(t)$. These power consumption models will be further addressed by the real measurements in Section V.

IV. Energy-Efficient SpeedControl Algorithm

In this section, we formulate an optimization problem considering energy minimization with queue stability for delay-tolerant applications, and develop an energy-minimal SpeedControl algorithm. At the end of this section, we demonstrate the theoretical analysis of the proposed algorithm.

A. Problem Formulation

Our objective for the queueing model shown in Fig. 1 is to develop energy-minimal joint application scheduling, CPU speed adjustment, network selection, and transmit power control policies with queue stability for delay-tolerant applications. The smartphone is able to serve all arrival workloads within the capacity region which is a set of all acceptable arrival rates with guaranteeing the stability of CPU and network queues. We formally state an optimization problem as follows:

$$(\mathbf{P}): \quad \min_{(\boldsymbol{\theta}, s, l, i)} \left(\lim_{T\to\infty} \frac{1}{T}\sum_{t=0}^{T-1}\left(P^c(s(t)) + P^n(l(t), i(t), t)\right)\right) \quad (5)$$

$$\text{s.t.} \limsup_{t\to\infty} \frac{1}{t}\sum_{\tau=0}^{t-1}\mathbb{E}\{Q^c(\tau) + Q^n(\tau)\} < \infty \quad (6)$$

where $Q^c(t) = Q^c_{\mathrm{NA}}(t) + Q^c_{\mathrm{NNA}}(t)$ and $(\boldsymbol{\theta}, s, l, i) \triangleq (\theta(t), s(t), l(t), i(t))_{t=0}^{\infty}$. The constraint (6) means that the average CPU and network queue lengths should be finitely maintained, i.e., all arrived workloads should be served within a finite time [28].

B. Algorithm Design

We obtain a solution of our problem **(P)** under unknown future sequences of wireless network states and workload arrivals

[6]A cellular network interface consumes tail energy after transmitting data for a few seconds. We will deal with this tail energy issue in Section V-B.

through invoking “Lyapunov drift-plus-penalty” method [28].

Making Single Objective. Our original objective is to minimize CPU and network power consumptions with queue stability in **(P)**. We first define Lyapunov function and Lyapunov drift function as follows:

$$L(t) \triangleq \frac{1}{2}\left[Q^c_{\mathrm{NA}}(t) + Q^n(t)\right]^2 + \frac{1}{2}\left[Q^c_{\mathrm{NNA}}(t)\right]^2 \quad (7)$$

$$\Delta(L(t)) \triangleq \mathbb{E}\{L(t+1) - L(t)|\mathbf{Q}(t)\} \quad (8)$$

where $\mathbf{Q}(t) = \{Q^c_{\mathrm{NNA}}(t), Q^c_{\mathrm{NA}}(t), Q^n(t)\}$. The Lyapunov function (7) is designed to fairly stabilize total NA queues $(Q^c_{\mathrm{NA}}(t) + Q^n(t))$ and NNA queues $(Q^c_{\mathrm{NNA}}(t))$. This is a key design principle to isolate the NNA performance from that of NA in terms of delay. Under this Lyapunov function design, if the network queues are accumulated due to the network bottleneck, scheduling NA would not reduce the NA queues $(Q^c_{\mathrm{NA}}(t) + Q^n(t))$, so the NNA would be always scheduled to reduce the NNA queues $(Q^c_{\mathrm{NNA}}(t))$. On the other hand, if workloads are not accumulated in the network queues, the NA and NNA would be fairly scheduled in terms of queue lengths.[7] This is just a criterion to isolate the performance of the NNA from that of NA as discussed in Section III.

Next, we define Lyapunov drift-plus-penalty function where the penalty function is the sum of expected CPU and network power consumptions during slot $t(\mathbb{E}\{P^c(s(t))|\mathbf{Q}(t)\} + \mathbb{E}\{P^n(l(t), i(t), t)|\mathbf{Q}(t)\})$ as follows:

$$\Delta(L(t)) + V\mathbb{E}\left\{P^c(s(t)) + P^n(l(t), i(t), t)|\mathbf{Q}(t)\right\} \quad (9)$$

where V is an energy–delay tradeoff parameter. Then, our single objective is to minimize the function (9).

Deriving Upper Bound. Next, we assume that workload arrival $A_{\mathrm{NA}}(t)$ and $A_{\mathrm{NNA}}(t)$, CPU speed $s(t)$, and uplink throughput $\mu(l(t), i(t), t)$ for all available networks and transmit power indices for all time slots are bounded as follows:

$$A_{\mathrm{NA}}(t) \leq A_{NA,max}, \quad A_{\mathrm{NNA}}(t) \leq A_{\mathrm{NNA,max}},$$
$$s(t) \leq s_{\max}, \quad \mu(l(t), i(t), t) \leq \mu_{\max}.$$

From the above bounds and queueing dynamics (1)–(3), the Lyapunov drift-plus-penalty function (9) is bounded as the following Lemma 1.

Lemma 1: Under any possible control variables $\theta(t) \in \{0, 1\}$, $s(t) \in \{s_1, s_2, \ldots, s_{\max}\}$, $l(t) \in B(t)$, and $i(t) \in \mathcal{I}_{l(t)}(t)$, we have:

$$\begin{aligned}
&\Delta(L(t)) + V\mathbb{E}\{P^c(s(t)) + P^n(l(t), i(t), t)|\mathbf{Q}(t)\} \\
&\leq B + V\mathbb{E}\{P^c(s(t)) + P^n(l(t), i(t), t)|\mathbf{Q}(t)\} \\
&\quad - \mathbb{E}\left\{Q^c_{\mathrm{NNA}}(t)\left(\frac{(1-\theta(t))s(t)}{\gamma_{\mathrm{NNA}}} - A_{\mathrm{NNA}}(t)\right)|\mathbf{Q}(t)\right\} \\
&\quad - \mathbb{E}\left\{(Q^c_{\mathrm{NA}}(t) + Q^n(t))\left(\min\left\{\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}} + Q^n(t),\right.\right.\right. \\
&\quad \left.\left.\left.\mu(l(t), i(t), t)\right\} - A_{\mathrm{NA}}(t)\right)|\mathbf{Q}(t)\right\} \quad (10)
\end{aligned}$$

where $B = \frac{1}{2}\left(\frac{s^2_{\max}}{\gamma^2_{\mathrm{NNA}}} + \mu^2_{\max} + A^2_{\mathrm{NA,max}} + A^2_{\mathrm{NNA,max}}\right)$.

[7]In fact, because the units of processing speed (in cycles/Δt) and traffic in the queues (in bits) are different, processing density (in cycles/bit) of the application should be also considered. In this case, however, we assume that processing densities of the NA and NNA are the same for ease of presentation.

Proof: Please refer to the Appendix. ■

Deriving Solution. Minimizing the left-hand side (LHS) of (10) means that our original problem **(P)** is satisfied. We demonstrate that the problem **(P)** has an optimal policy π^* at the following Theorem 1.

Theorem 1: For any mean arrival workload $\mathbb{E}\{A_{\mathrm{NA}}(t)\} = \lambda_{\mathrm{NA}}$ and $\mathbb{E}\{A_{\mathrm{NNA}}(t)\} = \lambda_{\mathrm{NNA}}$ within a capacity region, $\lambda_{\mathrm{NA}} + \lambda_{\mathrm{NNA}} \in \Lambda$,[8] there exists a stationary randomized control policy π^* that selects application scheduling $\theta(t)$, CPU speed $s(t)$, network interface $l(t)$ and transmit power index $i(t)$ every slot t while satisfying the following:

$$\mathbb{E}\{A_{\mathrm{NA}}(t)\} = \mathbb{E}\left\{\frac{\theta(t)^{\pi^*}s(t)^{\pi^*}}{\gamma_{\mathrm{NA}}}\right\} \quad (11)$$

$$\mathbb{E}\{A_{\mathrm{NNA}}(t)\} = \mathbb{E}\left\{\frac{(1-\theta(t)^{\pi^*})s(t)^{\pi^*}}{\gamma_{\mathrm{NNA}}}\right\} \quad (12)$$

$$\mathbb{E}\left\{\frac{\theta(t)^{\pi^*}s(t)^{\pi^*}}{\gamma_{\mathrm{NA}}}\right\} = \mathbb{E}\{\mu(l(t)^{\pi^*}, i(t)^{\pi^*}, t)\} \quad (13)$$

$$\begin{aligned}
&\mathbb{E}\{P^c(s(t)^{\pi^*})\} + \mathbb{E}\{P^n(l(t)^{\pi^*}, i(t)^{\pi^*}, t)\} \\
&\quad = \overline{P^c}(\lambda_{\mathrm{NA}} + \lambda_{\mathrm{NNA}}) + \overline{P^n}(\lambda_{\mathrm{NA}}) \quad (14)
\end{aligned}$$

where $\overline{P^c}(\lambda_{\mathrm{NA}} + \lambda_{\mathrm{NNA}}) + \overline{P^n}(\lambda_{\mathrm{NA}})$ is a minimum value of the sum of average CPU power and network power to process $\lambda_{\mathrm{NA}} + \lambda_{\mathrm{NNA}}$ and transmit λ_{NA}.

Proof: Please refer to the Appendix. ■

Then, an optimal algorithm (OPT) which minimizes (9) is to find control variables $(\theta(t), s(t), l(t), i(t))$ which minimize the LHS of (10) every slot, i.e., the optimal algorithm makes the right-hand side (RHS) of (10) be the smallest one among the values which obtain from all possible stationary randomized control policies. However, because the RHS of (10) is tightly coupled with all control variables, we design a simple and decoupled SpeedControl algorithm by making some approximations and assumptions as follows.

The RHS of (10) can be divided by three cases depending on two conditions. **(Case 1)** $\boldsymbol{Q^n(t) \geq \mu_{\max}(t)}$: The RHS of (10) in this case can be easily decomposed into subproblems to find $\theta(t) = 0$, $s(t)$ and $(l(t), i(t))$, respectively because $\min\{\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}} + Q^n(t), \mu(l(t), i(t), t)\}$ in the fourth line of (10) always becomes $\mu(l(t), i(t), t)$. **(Case 2)** $\boldsymbol{Q^n(t) < \mu_{\max}(t)}$: Since $\min\{\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}} + Q^n(t), \mu(l(t), i(t), t)\}$ does not always become $\mu(l(t), i(t), t)$, we make an approximation as follows:

$$Q^n(t) \approx 0, \quad \text{for } Q^n(t) < \mu_{\max}(t). \quad (15)$$

Then, (Case 2) can be divided by two subcases as follows. **(Case 2-1)** $\boldsymbol{\frac{Q^c_{\mathrm{NNA}}(t)}{\gamma_{\mathrm{NNA}}} \geq \frac{Q^c_{\mathrm{NA}}(t)}{\gamma_{\mathrm{NA}}}}$: The RHS of (10) in this case can be easily decomposed into subproblems to find $\theta(t) = 0$, $s(t)$ and $(l(t), i(t))$, respectively. Because $\theta(t)$ is 0 and we assume that $Q^n(t)$ is 0, the network part does not transmit data.

[8]Λ denotes all mean NA and NNA arrival workloads which the smartphone can transmit or process within a finite time.

(Case 2-2) $\boldsymbol{\frac{Q^c_{\text{NNA}}(t)}{\gamma_{\text{NNA}}} < \frac{Q^c_{\text{NA}}(t)}{\gamma_{\text{NA}}}}$: In order to decouple the RHS of (10) into subproblems, we make an assumption as follows:

$$\frac{s(t)}{\gamma_{\text{NA}}} \le \mu(l(t), i(t), t). \tag{16}$$

Then, the RHS of (10) can be decomposed into subproblems to find $\theta(t) = 1$, $s(t)$ and $(l(t), i(t))$, respectively.

Finally, the SpeedControl algorithm sequentially finds $\theta(t)$, $s(t)$, and $(l(t), i(t))$ for above three cases every slot t. Although we make some approximations and assumptions to derive the SpeedControl algorithm, this simple and decoupled algorithm demonstrates the similar energy saving and average delay performance with a complex optimal algorithm (OPT) in our simulation results (Section V-B).

C. SpeedControl Algorithm

The SpeedControl algorithm jointly controls application scheduling, CPU speed, network interface selection, and transmit power $(\theta^*(t), s^*(t), l^*(t), i^*(t))$ at each slot t described as follows.

SpeedControl Algorithm

```
For each time slot $t$,
 1: input :$(Q^c_{\text{NA}}(t), Q^c_{\text{NNA}}(t), Q^n(t), B(t), I_{l(t)}(t),$
        $\mu(l(t), i(t), t), P^n(l(t), i(t), t)$ for all $l(t)$ and $i(t))$
 2: if $Q^n(t) \ge \mu_{\max}(t)$ then
 3:   Schedule NNA $(\theta^*(t) = 0)$
 4:   Select CPU speed $s^*(t)$ by
        $\min_{s^*(t)} \left\{ VP^c(s(t)) - \frac{s(t)}{\gamma_{\text{NNA}}} Q^c_{\text{NNA}}(t) \right\}$   (17)
 5:   Select network $l^*(t)$ and transmit power index $i^*(t)$ by
        $\min_{l^*(t), i^*(t)} \{VP^n(l(t), i(t), t)$
            $-\mu(l(t), i(t), t)(Q^c_{\text{NA}}(t) + Q^n(t))\}$   (18)
 6: else if $Q^n(t) < \mu_{\max}(t)$ then
 7:   if $\frac{Q^c_{\text{NA}}(t)}{\gamma_{\text{NA}}} \ge \frac{Q^c_{\text{NNA}}(t)}{\gamma_{\text{NNA}}}$ then
 8:     Schedule NA $(\theta^*(t) = 1)$
 9:     Select CPU speed $s^*(t)$ by
        $\min_{s^*(t)} \left\{ VP^c(s(t)) - \frac{s(t)}{\gamma_{\text{NA}}} (Q^c_{\text{NA}}(t) + Q^n(t)) \right\}$   (19)
10:     Select $l^*(t)$ and $i^*(t)$ by
        $\min_{l^*(t), i^*(t)} P^n(l(t), i(t), t)$ s.t. $\frac{s^*(t)}{\gamma_{\text{NA}}} \le \mu(l(t), i(t), t)$   (20)
11:   else if $\frac{Q^c_{\text{NA}}(t)}{\gamma_{\text{NA}}} < \frac{Q^c_{\text{NNA}}(t)}{\gamma_{\text{NNA}}}$ then
12:     Schedule NNA $(\theta^*(t) = 0)$
13:     Select CPU speed $s^*(t)$ by (17)
14:     Do not select network $(l^*(t) = N)$
15:   end if
16: end if
17: output: $(\theta^*(t), s^*(t), l^*(t), i^*(t))$
```

where $\mu_{\max}(t)$ is the maximum uplink throughput among available networks at slot t.

The SpeedControl algorithm is divided by three cases depending on two conditions as follows.

(Case 1) $\boldsymbol{Q^n(t) \ge \mu_{\max}(t)}$: If the network queues become a bottleneck due to the fact that the networking speed is slower than processing speed, processing data at the CPU part does not reduce total queue lengths, hence always the NNA should be scheduled. Scheduling the NNA makes the average processing speed of NA be reduced, so the processing speed becomes similar with the average networking speed. Once the NNA is scheduled, the CPU speed is adjusted by the optimization problem (17). The first term with V of (17) can be interpreted as trying to minimize CPU power consumption, and second term without V can be explained as trying to minimize NNA CPU queue. An energy–delay tradeoff can be controlled by a single parameter V, e.g., as V becomes larger, the energy consumption gets lower by trading longer delay. After the CPU speed is selected, the network interface and transmit power are selected by the optimization problem (18). The controls of network part is similar with that of CPU part. The first term of (18) can be interpreted as trying to minimize the network power consumption, and the second term can be explained as trying to minimize total NA queues $(Q^c_{\text{NA}}(t) + Q^n(t))$.

(Case 2) $\boldsymbol{Q^n(t) < \mu_{\max}(t)}$ and $\boldsymbol{\frac{Q^c_{\text{NA}}(t)}{\gamma_{\text{NA}}} \ge \frac{Q^c_{\text{NNA}}(t)}{\gamma_{NNA}}}$: Because the network queues do not become a bottleneck from the fact that the networking speed is not slower than the processing speed, the NA and NNA should be fairly selected in terms of queue lengths. Thus, the application whose CPU queue lengths are longer, i.e., more urgent application, is selected when the processing densities of two applications are the same $(\gamma_{\text{NA}} = \gamma_{\text{NNA}})$. Consequently, the NA is selected in this case. Once the NA is scheduled, the CPU speed is adjusted by the optimization problem (19). After the CPU speed is selected, the network interface and transmit power are controlled by the optimization problem (20). The meaning of (20) is that the networking speed should be analogously controlled with the processing speed in case that the network queues do not become a bottleneck.

(Case 3) $\boldsymbol{Q^n(t) < \mu_{\max}(t)}$ and $\boldsymbol{\frac{Q^c_{\text{NA}}(t)}{\gamma_{\text{NA}}} < \frac{Q^c_{\text{NNA}}(t)}{\gamma_{\text{NNA}}}}$: In this case, the NNA is selected due to the similar reasons as **(Case 2)**. Because the NA is not scheduled, the network part does not transmit data in order to adjust the networking speed with the processing speed of NA.

D. Theoretical Analysis

The sum of NA and NNA queue lengths and the sum of average CPU and network power consumptions of an optimal algorithm (OPT) can be upper bounded by the following Theorem 2.

Theorem 2: Let $t = \{0, 1, \ldots T-1\}$. Suppose there exist $\epsilon' > 0$ and $\epsilon'' > 0$ such that $\lambda_{\text{NA}} + 2\epsilon' \in \Lambda_{\text{NA}}$ and $\lambda_{\text{NNA}} + \epsilon'' \in \Lambda_{\text{NNA}}$, then under the optimal algorithm (OPT), we have:

$$\limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{Q^c(t) + Q^n(t)\} \le \frac{B + VP^*(\epsilon, \epsilon)}{\epsilon} \tag{21}$$

$$\bar{P}^{c,\text{OPT}} + \bar{P}^{n,\text{OPT}} \le P^*(\epsilon, \epsilon) + \frac{B}{V} \tag{22}$$

where $Q^c(t) = Q^c_{\rm NA}(t) + Q^c_{\rm NNA}(t)$, $\epsilon = 2\epsilon' = \epsilon''$, $P^*(\epsilon, \epsilon) = P^{c^*}(\epsilon) + P^{n^*}(\epsilon)$ denotes the optimal lower bound of CPU and network power consumption.

Proof: Please refer to Appendix. ■

The result of Theorem 2 can describe the relationship of energy–delay tradeoff. As the energy–delay tradeoff parameter V, which can be controlled by user's preference, becomes smaller, the sum of average queue lengths (NA and NNA) gets smaller whereas the average power consumption gets larger. On the other hand, as V becomes larger, the average CPU and network power consumptions get smaller whereas the sum of average queue lengths gets larger. We verify these performance bounds at the simulation results in Section V-B.

V. Performance Evaluation

In this section, we evaluate the performance of the proposed algorithm through measurement, trace-driven simulations, and experiments.

A. Real Measurements and Traces

Real CPU Power Measurement. We measure and analyze the CPU power consumptions of five popular android smartphones for different CPU speeds. The five smartphones have different Kernels and OSs, so the CPU clock frequency-voltage matching tables of five smartphones are different, respectively. For example, Nexus S has six levels of CPU clock speeds, and each CPU speed matches with a specific voltage (100 MHz–975 mV, 200 MHz–975 mV, 400 MHz–1025 mV, 800 MHz–1250 mV, 1000 MHz–1450 mV, 1440 MHz–1500 mV).

We connect Monsoon power monitor [33] to the five smartphones and measure the power consumption. Because the power consumption of a CPU module cannot be directly measured, we turn off the other components such as WiFi, cellular, Bluetooth, GPS modules and kill the running tasks using well known task killer application [34] except for a video transcoding application[9] [35] for 100% utilization of CPU.

Fig. 2 depicts the measured CPU power consumptions as a function of CPU speeds for five different smartphones. The points (star, triangle, circle, and square) denote the real power measurement values for different discrete CPU speed levels. The measured discrete power consumptions are well modeled by a cubic polynomial scaling of speed to power for all smartphones. Interestingly, it has different polynomial with that of typical laptop processor or TCP offload engine [16] where the polynomial is closer to the quadratic. From these results, we design a CPU power consumption model as a function of the CPU speeds as follows:

$$P^c(s(t)) = \alpha s(t)^3 + \beta \tag{23}$$

where $s(t)$ denotes CPU speed, t denotes time slot whose interval is the minimum static CPU speed period, α and β denote constant values which are different from each smartphone.

Real Network Power Measurement. We measure the power consumption of 3G and WiFi network interfaces using Monsoon power monitor [33] for two different android smartphones, respectively. For network interface measurement, we turn off

[9]In our measurement, all smartphones have 100% utilization for all CPU speed levels when running video trascoding application. Also, because we run one application, only one CPU core can be operated in spite of dual core CPU.

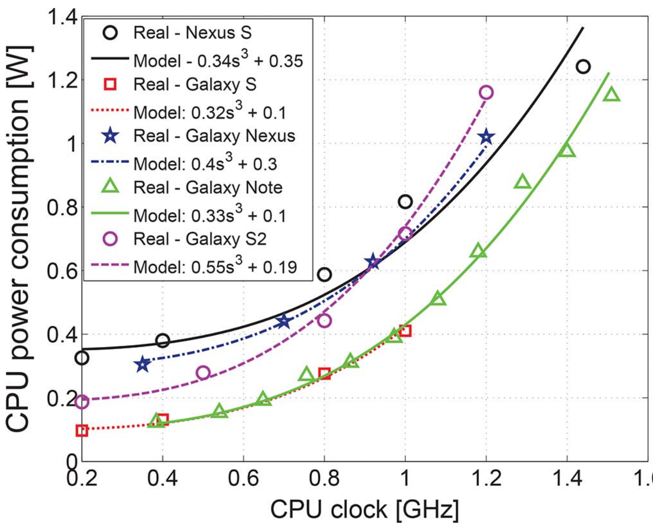


Fig. 2. CPU power consumption as a function of CPU speed.

running applications in the smartphone except for an application which is made by us for transmitting dummy data to a server. We also develop an application which measures uplink throughput in the server. Then, we measure the power consumption of the smartphone using Monsoon power monitor and measure the uplink throughput in the server while the smartphone transmits 2 MByte (WiFi) and 50 KByte (3G) of dummy data to the server.

The measured network interface power consumption of the two smartphones are shown in Table II, which provides the following findings: *(i)* the transmit powers of 3G and WiFi interfaces are similar in both smartphones, yet *(ii)* WiFi transmit power in W/bit is much smaller than that of 3G. This implies that the WiFi networks are more energy efficient than 3G networks for transmitting the same amount of data. *(iii)* the average transmit power consumption in W/bit of WiFi and 3G for two devices are comparable with CPU power consumption in W/bit of two devices (CPU power in W/bit assuming that CPU speed is 1 GHz and processing density is 1000 cycles/bit[10]: 816.5×10^{-9} W/bit (Nexus S), 688.1×10^{-9} W/bit (Galaxy Nexus)). It supports the fact that the power managements of CPU and network interfaces are equally important.

Real Traces. We collect workload arrivals, processing densities of two kinds of applications (NAs and NNAs) and uplink throughputs of 3G and WiFi networks. First, we use real YouTube video data size distribution from [36] to generate workload arrival traces of the two kinds of applications. Second, we run a video transcoding application [35] in a smartphone for several video clips and measure their completion times, respectively. Then, we compute processing densities by dividing the processing quantity (in cycles) with the size of a video clip (in bits). The range of measured processing densities is between 200 cycles/bit to 1200 cycles/bit for different video formats and clips. Third, we measure the 3G and WiFi uplink throughputs and WiFi connectivities of five smartphones for two weeks in metropolitan areas (Seoul and Daegu) of South Korea. Using our uplink measurement application, the smartphones transmit dummy traffic to the 3G and WiFi networks every 20 seconds, respectively, and records the WiFi connectivity logs. Then,

[10]This value is calculated based on the real measurements of CPU utilization and processing time of the application.

a server application measures the uplink throughput of all smartphones. The measured average uplink throughputs for 3G and WiFi are 0.76 Mbps and 3.01 Mbps, respectively, and the average WiFi temporal coverage is 63% in daytime (9:00 AM to 9:00 PM).

B. Trace-Driven Simulation

In this subsection, we verify our SpeedControl algorithm by trace-driven simulations under various environments.

Setup. We consider a scenario that a smartphone runs two applications: *(i)* In networking application (NA), a video clip generated in the smartphone is transcoded and transmitted to the cloud server. *(ii)* In non-networking application (NNA), the other video clip generated in the smartphone is just transcoded. During 1 second, the video clips where the sizes are $A_{\rm NA}(t)$ and $A_{\rm NNA}(t)$, are independently generated with 0.8% probability. Also, the smartphone gets around in some places which have coverages of WiFi and 3G networks. Assume that the WiFi networks are intermittently available, but the 3G networks are always available.[11] Among available networks, the smartphone can select one network interface for transmitting data or not. We assume that the WiFi interface of the smartphone is able to control transmit power and MCS level (i.e., data rate) for 10^{-5} BER target. So, we use data rate levels of 802.11a/g standards[12] and required SNR table for 10^{-5} BER target in [24]. We use the measured uplink throughput and WiFi connectivity distribution in our simulation.

For the power consumption profiles, we use one of the CPU power-speed sets from Fig. 2, and use the maximum transmit and idle powers for the 3G and WiFi interfaces from Table I. The control intervals are 1 second for the application scheduling, CPU speed adjustment, and transmit power control, and 20 seconds for the network selection. This is a reasonable setting because the associated networks cannot be changed as fast as the CPU speed adjustment due to the vertical handover delay. For the uplink throughput estimation in the simulations,[13] we assume that the current achievable uplink throughput is the same as that of previous time slot when the same transmit power is used. Then, by using this (transmit power, uplink throughput) combination, we make new feasible (transmit power, data rate) combination set. If the smartphone did not transmit data through the same network at the previous time slot, it uses the average (transmit power, uplink throughput) combination of the network as a current value.

Performance metrics are the average CPU and/or network energy consumptions per video clip and the average delay of the two kinds of applications per video clip.[14]We compare DVFS + SALSA and Max + SALSA with our SpeedControl

TABLE I
NETWORK POWER CONSUMPTIONS FOR TWO SMARTPHONES

Nexus S				
	idle(mW)	transmit(mW)	uplink throughput(Mbps)	average tx. power in W/bit
WiFi	230	702±72.5	1.66-3.12	308.6×10^{-9}
3G	213	1217±185	0.69-0.85	1700×10^{-9}

Galaxy Nexus				
	idle(mW)	transmit(mW)	uplink throughput(Mbps)	average tx. power in W/bit
WiFi	99	875±22	6.36-6.87	134×10^{-9}
3G	82	964±210	0.354-0.742	1951×10^{-9}

algorithm. DVFS + SALSA is conventional DVFS[15] with delayed network selection [20], and Max+SALSA uses always maximum CPU speed with delayed network selection.

Observations. From the simulation results, we obtain interesting observations as follows.

Relation with the energy–delay tradeoff parameter *V*. Fig. 3 depicts the average power consumption and queue lengths as a function of V, which can be controlled by user's preference. As described in Section IV-D and depicted in Fig. 3, the bound of average sum of NA and NNA queue lengths is linearly proportional to V in (21), and the bound of average sum of CPU and network power consumptions is inversely proportional to V in (22). However, because V has no physical meaning, from now on, we demonstrate the direct relations between the average power consumption and delay performance.

Energy–delay tradeoff. Fig. 5 depicts the energy–delay tradeoff for several algorithms. *(i)* It is worthy of notice that most of CPU energy saving and total energy saving (52% CPU, 51% total) can be obtained by trading only 10 minutes delay. This is because the CPU power consumption is modeled by a cubic polynomial scaling of CPU speed such as (23). Therefore, by smoothing the CPU speed along with slot t, CPU power can be saved. However, because smoothing the CPU speed makes the system be insensitive to the queue variation, average delay would be longer. Network power consumption can be saved until 50% by trading 10 minutes delay. This power saving comes from that the smartphone is reluctant to transmit data through 3G (or bad channel state and low throughput) which is energy-inefficient network than WiFi; yet it would wait for WiFi networks (or good channel state and high throughput). Although waiting for WiFi networks is good for energy saving, longer waiting than a certain delay constraint, e.g., 15 minutes, can hardly make the smartphone save more energy. *(ii)* SpeedControl (SC) saves 52%, 48% (in CPU), 42%, 43% (in total) energy for 10 minutes average delay than Max + SALSA and DVFS + SALSA, respectively. The power saving gains of SpeedControl come from that the algorithm pushes NA workloads from CPU side to network side only when the network side requires the workloads. This implies that joint consideration of application scheduling, CPU speed, network selection and transmit power control is imperative for optimizing CPU and network power in a smartphone. *(iii)* SpeedControl well catches up with the performance of the optimal algorithm even though SpeedControl is much low complex algorithm than the optimal algorithm.

[11]We have adopted 3G as a representative of the cellular technology because of ease of availability. The same experiments for our algorithm without any modification can be carried out for other cellular technologies, e.g., 4 G or LTE.

[12]A recent 802.11n standard [37] has more MCS indices due to more modulations, coding rates and bandwidths, so the impact of transmit power control on the energy saving would increase.

[13]We also use this estimation method in experiment

[14]We consider the average delay instead of the average queue lengths in the most of simulation results for practical analysis. By Little's law [38], the avearage delay can be interpreted as the average queue lengths divided by the average arrival rate.

[15]In this algorithm, the CPU speed is set to maximum when CPU workloads are greater than a threshold, and linearly decrease when the CPU workloads are less than the threshold which can be manually controllable [18].

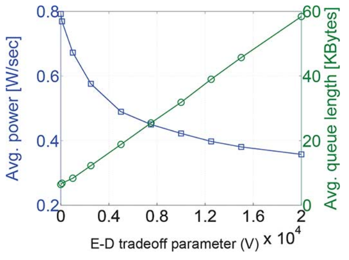


Fig. 3. Average power and queue length as a function of V.

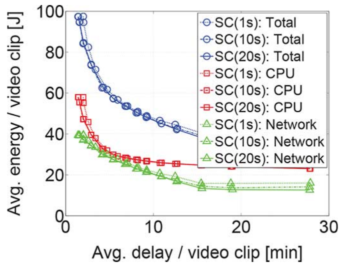


Fig. 4. Impact of time scale difference.

Impact of no transmission option and transmit power control. Fig. 6 demonstrates the impact of no transmission option and transmit power control on the energy–delay performance. *(i)* As V becomes larger, i.e., more average delay is allowed, SpeedControl (SC) has more energy saving gain than SpeedControl without no transmission option (i.e., the smartphone is able to select only 3G or WiFi networks, SC w/o NoTx) and SpeedControl without transmit power control (i.e., the smartphone always uses the maximum transmit power, SC w/o TPC). This is because that no transmission option enables the device to wait for WiFi networks which are more energy-efficient networks than 3G, and the transmit power control enables the device to exploit opportunism of time-varying wireless channel condition, e.g., the transmit power can be reduced when the wireless channel condition is bad. *(ii)* SpeedControl saves 23% and 17% of total energy consumptions when compared with SC w/o NoTx and SC w/o TPC by trading 10 minutes delay. The reason why the impact of energy saving gain in case of the no transmission option is much higher than that of the transmit power control is that the throughput gap between 3G and WiFi is greater than that of the wireless channel variation in the WiFi networks.

Impact of time scale difference. We verify the impact of the time scale difference between CPU speed control (or transmit power control) and network selection. Fig. 4 depicts the energy–delay tradeoff of SpeedControl for several time scales of the network selection, i.e., the network interface is selected every 1 second, 10 seconds and 20 seconds. This figure demonstrates that the average energy consumption and delay performance of SpeedControl for different network selection time scales are almost the same.[16] This means that practically selecting network interface every 20 seconds scarcely affects the performance degradation.

Impact of application scheduling. To verify that SpeedControl isolates the performance of the NA from that of NNA in terms of delay, we run the simulation in a network bottleneck case (i.e., WiFi temporal coverage: 0%, average network

[16]Because the estimated throughput is not exactly same with real achievable throughput, SC(20s) may have a chance to achieve better performance than others.

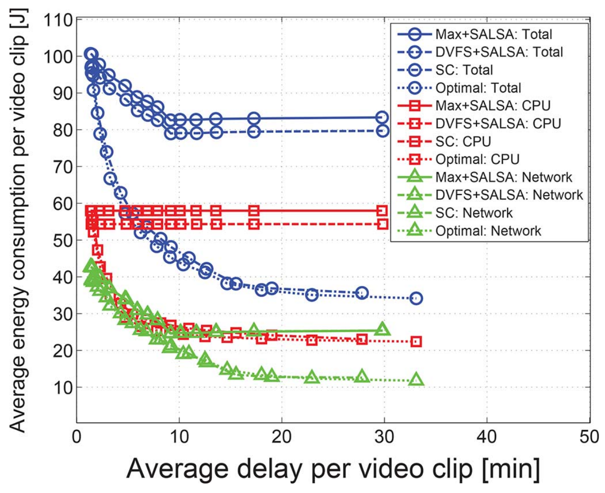


Fig. 5. Energy–delay tradeoff: WiFi temporal coverage—65%, processing density $(\mathrm{NA}, \mathrm{NNA}) = (300$ cycles/bit,1200cycles/bit).

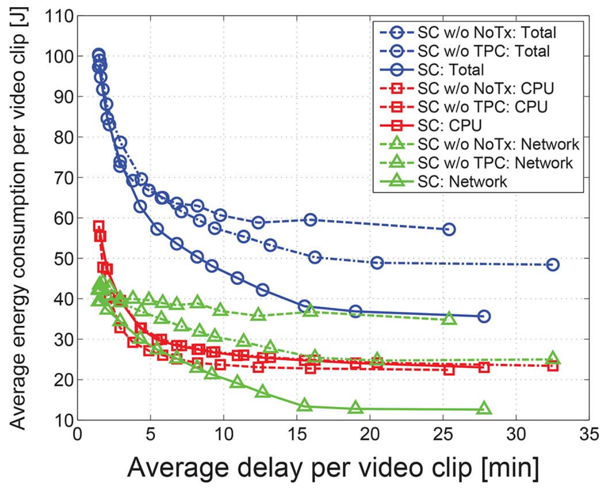


Fig. 6. Impact of TPC and NoTx: WiFi temporal coverage -65%, $(\gamma_{\mathrm{NA}}, \gamma_{\mathrm{NNA}}) = (300$ cycles/bit, 1200 cycles/bit).

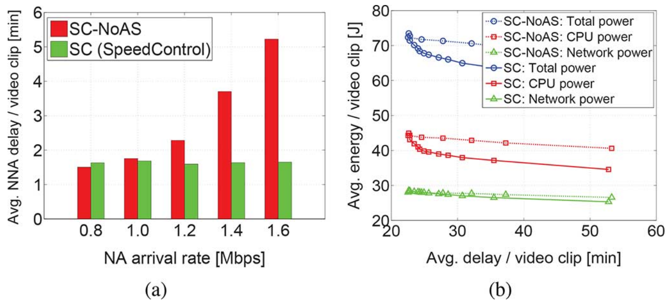


Fig. 7. Impact of application scheduling in network bottleneck case: WiFi temporal coverage—0%, average network throughput = 0.76 Mbps, $(\gamma_{\mathrm{NA}}, \gamma_{\mathrm{NNA}}) = (600$ cycles/bit, 600 cycles/bit). (a) Average delay of NNA. (b) Energy–delay tradeoff.

throughput: 0.76 Mbps). For comparison, we consider an algorithm without application scheduling control, called SC-NoAS. This algorithm schedules the application by first in first out (FIFO) manner, but the CPU speed adjustment and network selection[17] are operated like SpeedControl.

[17]Because the WiFi temporal coverage is zero in this simulation, we did not consider the transmit power control.

Fig. 7(a) and (b) depict delay performance of NNA and energy–delay tradeoff of SC-NoAS and SpeedControl when the network queues become a bottleneck. In the simulation for Fig. 7(a), the arrival rate of NNA is the same (0.8 Mbps), but the arrival rate of NA increases up to twofold. Fig. 7(a) demonstrates that SpeedControl guarantees the average delay of NNA when the arrival rate of NA increases, whereas the SC-NoAS increases average delay of NNA even though the arrival rate of NNA does not increase. This implies that SpeedControl makes NNA do not be influenced by the network environment, which is also related with a design issue of SpeedControl mentioned in Section IV-B. As a result, SpeedControl achieves better performance than SC-NoAS as shown in Fig. 7(b).

Impact of processing density, arrival rate and WiFi temporal coverage. Fig. 8 depicts total $(\text{CPU} + \text{network})$ energy consumptions of existing algorithms $(\text{DVFS} + \text{SALSA}, \text{Max} + \text{SALSA})$, normalized by total energy consumptions of SpeedControl as a function of the sum of average NA and NNA delay per video clip. For this simulation, we generate WiFi temporal coverage traces for different WiFi coverages using measured WiFi temporal coverage distribution and uplink throughput.

(i) As the processing density of NA becomes smaller, and the arrival rate of NA gets higher, SpeedControl obtains more energy saving gain for more than 5 minutes delay. The gain from lower processing density of NA comes from that SpeedControl more quickly responds to the needs of network side than higher processing density case. The gain from higher arrival rates of NA comes from that SpeedControl knows when the network side requires the workloads to transmit whereas DVFS + SALSA and Max + SALSA do not know what happens in the network side. *(ii)* As the arrival rates of NNA become smaller, SpeedControl incurs more energy saving gain. This implies that the NA gives more energy saving impact on the smartphone than NNA due to that the NA uses both processing and networking resources whereas the NNA uses only processing resources. *(iii)* As the WiFi temporal coverage becomes wider, SpeedControl achieves more energy saving for more than 5 minutes delay. This is due to that the wider WiFi coverage enables the smartphone users to exploit more energy-efficient networks, i.e., WiFi networks; thus, the average networking power is reduced, which means that the CPU power consumption has a relatively bigger influence on total energy consumptions. Because our SpeedControl algorithm obtains relatively higher energy saving gain in the CPU part than network part as shown in Fig. 5, a total energy savings of SpeedControl increases when compared with the other algorithms as the WiFi temporal coverage extends.

Practical issues for energy consumption. We deal with two practical issues for energy consumption of wireless interfaces. First, 3G interface consumes tail energy for a few seconds after completing data transmission in practice [39]. To confirm this, we run the simulation to explore the impact of tail energy under the various Vs and normalized file sizes where the average reference file size is 6.249 MBytes. As shown in Fig. 9, the smartphone consumes a considerable tail energy (maximum 26% of total networking energy) depending on the V and normalized file size. We refer readers to [21], [39] for an optimization of the tail energy.

Second, to confirm how much WiFi on/off delays affect the delay performance of the NA and NNA, we measure the WiFi on/off delays and WiFi contact traces. The measured average WiFi on delay is 1.5–2 seconds and off delay is 0.5 seconds. Furthermore, our measurement results of WiFi contact traces and [22] demonstrate that the average WiFi inter-contact time is about 2 hours, and the energy efficiency of WiFi is higher than that of 3G. These features make the smartphone infrequently change the network interface, thus the interval of network interface transition would be much longer as compared to the WiFi on/off delays. From these observations, the WiFi on/off delays scarcely exert influence on the delay performance of SpeedControl algorithm in the real mobile environment.

C. Experiment

Setup. We develop a prototype of SpeedControl application which adopts our SpeedControl algorithm using Android software development kit (SDK) based on the open source code of NSTools application [40] which enables a smartphone to control CPU clock manually. The SpeedControl application determines the CPU speed and network selection (between no transmission and WiFi) based on the CPU and network queues and the estimation of current uplink throughput. For estimation of the uplink throughput, our private server transmits acknowledgement (ACK), which contains the uplink throughput information, to device every 5 seconds. Then, the smartphone considers received ACK as the current throughput. If the device does not transmit data in previous time slot, the average uplink throughput is considered as the current throughput. For experiments, we prepare rooted smartphone[18] (Nexus S) and Monsoon power monitor [33].

The experimental smartphone runs two applications: *(i)* video transcoding application, *(ii)* prototype of our SpeedControl application which selects the CPU speed and network interface and then transmit the transcoded data to our server, yet they can be seen as one networking application. We assume that the trascoding speed is the same as CPU speed. Then, the transcoded data is transmitted from the network queues to our private server. We consider a situation that 5 video clips (21 MByte per one clip) are arrived at the specific times, respectively. Also, the smartphone is associated with one WiFi AP.[19] and connected to Monsoon power monitor to measure the energy consumption. As performance metrics, we measure *(i)* battery level by the application (visualized as % or bar in typical smartphones) and real power using Monsoon power monitor and *(ii)* the average delay of video clips when four video clips are fully transmitted.

Observation. We obtain three observations from experimental results of the SpeedControl, Max + SALSA and DVFS + SALSA algorithms. *(i)* (Fig. 10(a)) The DVFS + SALSA and Max + SALSA consume about 70% and 80% more energy than the SpeedControl algorithm with the same delay (about 11 minutes) in real power measurement for transmitting 4 video clips. *(ii)* (Fig. 10(b)) Our SpeedControl consumes 50% and 40% less battery than Max + SALSA and

[18]To manually control CPU clock frequencies, the smartphone SO attains privileged control within the subsystem of Android.

[19]This WiFi AP is a private AP for only one experimental smartphone.

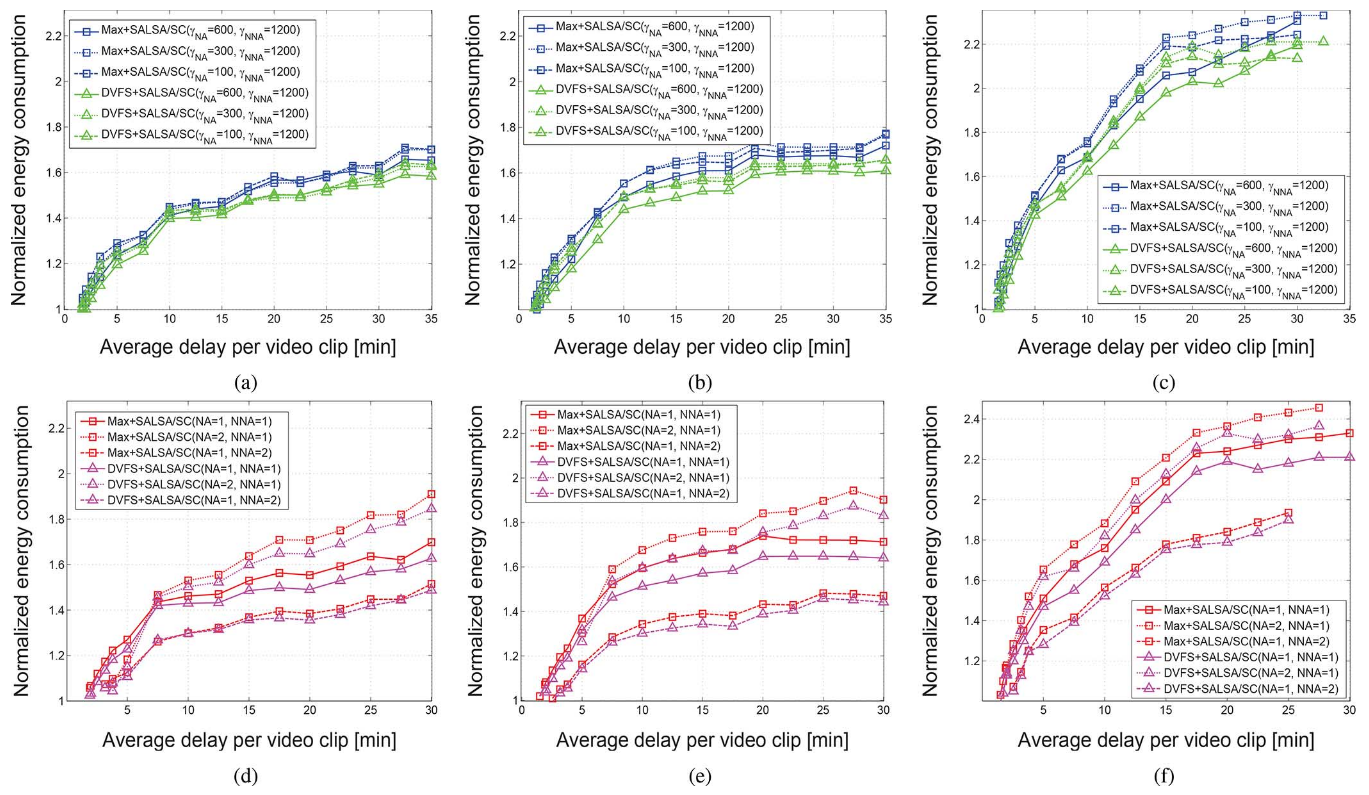


Fig. 8. Normalized energy consumptions for various WiFi temporal coverages: (a)–(c) are the results under the same arrival rate $(\mathrm{NA} : \mathrm{NNA} = 1 : 1)$, (d)–(e) are the results under the same processing density $(\gamma_{\mathrm{NA}} : \gamma_{\mathrm{NNA}} = 300$ cycles/bit:1200 cycles/bit). (a) WiFi temporal coverage: 35%. (b) WiFi temporal coverage: 50%. (c) WiFi temporal coverage: 65%. (d) WiFi temporal coverage: 35%. (e) WiFi temporal coverage: 50%. (f) WiFi temporal coverage: 65%.

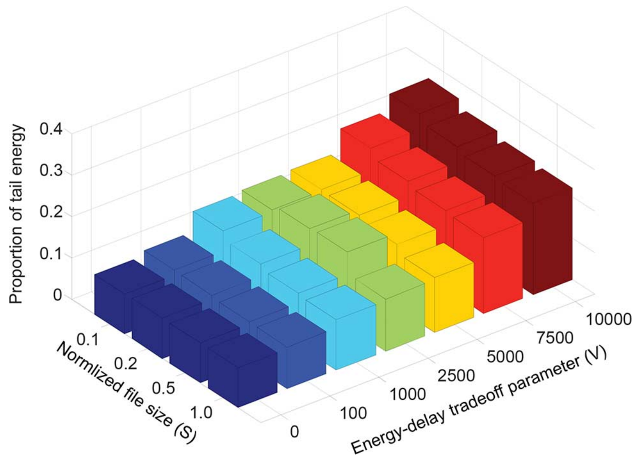


Fig. 9. Proportion of tail energy (tail energy/total networking energy): WiFi temporal coverage—35%, $(\gamma_{\mathrm{NA}}, \gamma_{\mathrm{NNA}}) = (300, 1200)$ cycles/bit.

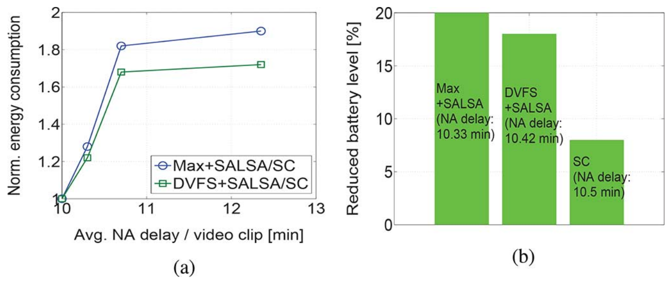


Fig. 10. Experimental results for SpeedControl, $\mathrm{Max}+\mathrm{SALSA}$ and $\mathrm{DVFS}+\mathrm{SALSA}$. (a) Energy–delay tradeoff. (b) Reduced battery level.

DVFS + SALSA by trading similar delay. *(iii)* Smartphone users who install our SpeedControl application save 10% of battery level (spend 10% of battery) for uploading 4 video clips (total 84 MBytes) by trading about 3 minutes more delay when the starting battery level is 70%.

VI. Conclusion

In this paper, we investigated key processing and networking features of contemporary smartphones in terms of tradeoff between energy consumption and application delay. Based on this study, we suggest a SpeedControl algorithm, which jointly optimizes CPU speed, wireless interface selection and transmit power so as to answer how much energy can be saved further by the joint optimization when applications can tolerate a certain delay. SpeedControl isolates the performance of non-networking applications from that of networking applications as well as obtains high energy saving by trading small delay. Finally, through extensive simulations and experiment studies including meaningful real measurement results such as smartphone power consumption or network states, we made several important observations which provide us with a message that joint optimization of CPU and network speed would be imperative, especially in future network trend where the more energy-efficient networks are deployed. Furthermore, the joint optimization of processing speed and networking speed for energy minimization might be applied for emerging multi-homed terminals and real-time video/multimedia applications.

APPENDIX A

A. Proof of Lemma 1

Proof: From the queueing models of networking application (NA),

$$Q^c_{\mathrm{NA}}(t+1)+Q^n(t+1) = [Q^c_{\mathrm{NA}}(t)+Q^n(t)-\mu'(t)+A_{\mathrm{NA}}(t)]^+ \quad (24)$$

where $\mu'(t)=\min\{\mu(l(t),i(t),t),Q^n(t)+\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}}\}$ since the actual service quantity of network part is bounded by actual queue length at time slot t. By taking square on the (24) and using the fact that $([X]^+)^2 \le X^2$, we have

$$\begin{aligned}(Q^c_{\mathrm{NA}}(t+1)+Q^n(t+1))^2 &\le (Q^c_{\mathrm{NA}}(t)+Q^n(t)-\mu'(t)+A_{\mathrm{NA}}(t))^2\\ &\le (Q^c_{\mathrm{NA}}(t)+Q^n(t))^2-2(\mu'(t)-A_{\mathrm{NA}}(t))\\ &\quad\times(Q^c_{\mathrm{NA}}(t)+Q^n(t))+\mu^2_{\max}+A^2_{\mathrm{NA},\max}. \quad (25)\end{aligned}$$

Then, by arranging (25), we have

$$\begin{aligned}&(Q^c_{\mathrm{NA}}(t+1)+Q^n(t+1))^2-(Q^c_{\mathrm{NA}}(t)+Q^n(t))^2\\ &\le -2(\mu'(t)-A_{\mathrm{NA}}(t))(Q^c_{\mathrm{NA}}(t)+Q^n(t))\\ &\quad+\mu^2_{\max}+A^2_{\mathrm{NA},\max}. \quad (26)\end{aligned}$$

Similarly, we obtain the following by repeating for non-networking application (NNA):

$$\begin{aligned}&Q^c_{\mathrm{NNA}}(t+1)^2-Q^c_{\mathrm{NNA}}(t)^2\\ &\le -2\left(\frac{(1-\theta(t))s(t)}{\gamma_{\mathrm{NNA}}}-A_{\mathrm{NNA}}(t)\right)Q^c_{\mathrm{NNA}}(t)\\ &\quad+\frac{s^2_{\max}}{\gamma^2_{\mathrm{NNA}}}+A^2_{\mathrm{NNA},\max}. \quad (27)\end{aligned}$$

By summing over (26), (27) and CPU and network power consumption, we obtain the upper bound of following Lyapunov drift plus penalty function.

$$\begin{aligned}&\Delta(L(t))+V\mathbb{E}\{P^c(s(t))+P^n(l(t),i(t),t)|\mathbf{Q}(t)\}\\ &\le B+V\mathbb{E}\{P^c(s(t))+P^n(l(t),i(t),t)|\mathbf{Q}(t)\}\\ &\quad-\mathbb{E}\left\{\left(\min\left\{\mu(l(t),i(t),t),\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}}+Q^n(t)\right\}\right.\right.\\ &\quad\left.-A_{\mathrm{NA}}(t))(Q^c_{\mathrm{NA}}(t)+Q^n(t))|\mathbf{Q}(t)\right\}\\ &\quad-\mathbb{E}\left\{\left(\frac{(1-\theta(t))s(t)}{\gamma_{\mathrm{NNA}}}-A_{\mathrm{NNA}}(t)\right)Q^c_{\mathrm{NNA}}(t)|\mathbf{Q}(t)\right\}\end{aligned} \quad (28)$$

where $B=\frac{1}{2}(\frac{s^2_{\max}}{\gamma^2_{\mathrm{NNA}}}+\mu^2_{\max}+A^2_{\mathrm{NA},\max}+A^2_{\mathrm{NNA},\max})$. This completes the proof. ■

B. Proof of Theorem 1

Proof: Necessary condition is trivial. Therefore, we will prove the sufficient condition for (11)–(14).

Proof of (11) and (12) Let $\mathcal{C}=\mathcal{C}_{\mathrm{NA}}+\mathcal{C}_{\mathrm{NNA}}$ be the convex hull of the service rates from CPU queue and $C\in\mathcal{C}$. Then, by Caratheodory's theorem [41], C is decomposed into a convex combination of service rate corresponding to all combinations of control parameters $s(t)\in\{s_1,\ldots,s_{\max}\}$ and $\theta(t)\in\{0,1\}$ as follows:

$$\begin{aligned}C&=p^{(s_1,\theta=1)}\frac{s_1}{\gamma_{\mathrm{NA}}}+\cdot+p^{(s_{\max},\theta=1)}\frac{s_{\max}}{\gamma_{\mathrm{NA}}}\\ &\quad+p^{(s_1,\theta=0)}\frac{s_1}{\gamma_{\mathrm{NNA}}}+\cdots+p^{(s_{\max},\theta=0)}\frac{s_{\max}}{\gamma_{\mathrm{NNA}}}\\ &=\mathbb{E}\left\{\frac{\theta(t)s(t)}{\gamma_{\mathrm{NA}}}\right\}+\mathbb{E}\left\{\frac{(1-\theta(t))s(t)}{\gamma_{\mathrm{NNA}}}\right\}. \quad (29)\end{aligned}$$

Since the probability vector $(p^{(s_1,\theta=1)},\ldots,p^{(s_{\max},\theta=0)})$ can be represented by $\{s(t),\theta(t)\}_{t=0}^{\infty}$ sequences by a law of large numbers, there exist randomized control policies which support any arrival rate $\lambda_{\mathrm{NA}}+\lambda_{\mathrm{NNA}}$ within capacity region. It means that there exist randomized $\{s(t),\theta(t)\}$ control policies every time slot t which satisfies (11) and (12).

Proof of (13). Let $g\in G$ be the wireless channel state of networks such as cellular or WiFi. Also, let $\mathcal{C}^{\mathrm{NA}}_g$ be the convex hull of the service rate from network queue when the channel state is g and $(C^{\mathrm{NA}}_{g_1},\ldots C^{\mathrm{NA}}_{g_{\max}})\in\mathcal{C}^{\mathrm{NA}}$. Then, by Caratheodory's Theorem [41], C^{NA} is decomposed into a convex combination of service rate corresponding to all combinations of control parameters $l(t)\in\{l_1,\ldots l_{\max}\}$, $i(t)\in\{i_1,\ldots,i_{\max}\}$ and the channel states $g\in\{g_1,\ldots,g_{\max}\}$ as follows:

$$\begin{aligned}C^{\mathrm{NA}}&=p^{(l_1,i_1)}_{g_1}\mu(l_1,i_1,t)+\cdots\\ &\quad+p^{(l_{\max},i_{\max})}_{g_{\max}}\mu(l_{\max},i_{\max},t)\\ &=\sum_{g\in G}\pi_g\mathbb{E}\{\mu(l(t),i(t),t)|g(t)=g\}. \quad (30)\end{aligned}$$

Then, intuitively the time average of the $\mu(l(t),i(t),t)$ process can be expressed as a sum over the steady-state probabilities as follows:

$$\begin{aligned}&\lim_{t\to\infty}\frac{1}{t}\sum_{\tau=0}^{t-1}\mu(l(t),i(t),t)\\ &=\sum_{g\in G}\pi_g\mathbb{E}\{\mu(l(t),i(t),t)|g(t)=g\}\\ &=\mathbb{E}\{\mu(l(t),i(t),t)\} \quad (31)\end{aligned}$$

where π_g is the steady-state probability of g. Since the probability vector $(p^{(l_1,i_1)}_{g_1},\ldots,p^{(l_{\max},i_{\max})}_{g_{\max}})$ can be represented by $\{l(t),i(t)\}_{t=0}^{\infty}$ sequence by a law of large number, there exist randomized control policies which support any arrival rate λ_{NA} within capacity region. It means that there exist randomized $\{l(t),i(t)\}$ control policies every time slot t which satisfies (13).

Proof of (14). Among all control sequences $(s(t),\theta(t),l(t),i(t))_{t=0}^{\infty}$ to satisfy (11), (12), and (13), we can find an optimal sequence π^* to minimize the sum of average CPU power and network power, and it satisfies (14). This completes the proof. ■

C. Proof of Theorem 2

Proof: First, we prove average queue bound (21). Since $\lambda_{\mathrm{NA}}+2\epsilon'\in\Lambda_{\mathrm{NA}}$ and $\lambda_{\mathrm{NNA}}+\epsilon''\in\Lambda_{\mathrm{NNA}}$, following relationships can be shown using Theorem 1 that there exists a stationary and randomized policy $\pi^{*'}$.

$$\mathbb{E}\{P^{\pi^{*'}}(t)\} = P^*(2\epsilon' + \epsilon'') \quad (32)$$

$$\mathbb{E}\{A_{\text{NA}}(t)\} = \mathbb{E}\left\{\frac{\theta(t)^{\pi^{*'}} s(t)^{\pi^{*'}}}{\gamma_{\text{NA}}}\right\} - \epsilon' \quad (33)$$

$$\mathbb{E}\{A_{\text{NNA}}(t)\} = \mathbb{E}\left\{\frac{(1-\theta(t)^{\pi^{*'}}) s(t)^{\pi^{*'}}}{\gamma_{\text{NNA}}}\right\} - \epsilon'' \quad (34)$$

$$\mathbb{E}\left\{\frac{\theta(t)^{\pi^{*'}} s(t)^{\pi^{*'}}}{\gamma_{\text{NA}}}\right\} = \mathbb{E}\{\mu(l(t)^{\pi^{*'}}, i(t)^{\pi^{*'}}, t)\} - \epsilon' \quad (35)$$

where $P^*(2\epsilon', \epsilon'') = P^{c*}(\lambda_{\text{NA}} + 2\epsilon', \lambda_{\text{NNA}} + \epsilon'') + P^{n*}(\lambda_{\text{NA}} + 2\epsilon')$ is the optimal average sum of CPU and network power for the average arrival rate $(\lambda_{\text{NA}} + 2\epsilon', \lambda_{\text{NNA}} + \epsilon'')$. By applying the above equations (32)–(35) in Lemma 1, we have

$$\begin{aligned}\Delta(L(t)) &+ V\mathbb{E}\{P^{c,OPT}(s(t)) + P^{n,\text{OPT}}(l(t), i(t), t)|\mathbf{Q}(t)\} \\ &\leq B + VP^*(2\epsilon', \epsilon'') - \mathbb{E}\{2\epsilon'\left(Q^c_{\text{NA}}(t) + Q^n(t)\right)|\mathbf{Q}(t)\} \\ &- \mathbb{E}\{\epsilon'' Q^c_{\text{NNA}}(t)|\mathbf{Q}(t)\}. \end{aligned} \quad (36)$$

By taking expectations on both sides of (36),[20] we have

$$\begin{aligned}\mathbb{E}\{L(t+1) - L(t)\} &+ 2\epsilon'\mathbb{E}\left\{(Q^c_{\text{NA}}(t) + Q^n(t))\right\} \\ &+\epsilon''\mathbb{E}\left\{Q^c_{\text{NNA}}(t)\right\} \leq B + VP^*(2\epsilon', \epsilon''). \end{aligned} \quad (37)$$

Without loss of generality, by taking $2\epsilon' = \epsilon'' = \epsilon$ and dividing both sides of (37) with ϵ and summing over $t = \{0, 1, \ldots, T - 1\}$, we have

$$\begin{aligned}\mathbb{E}\{L(T) - L(0)\} &+ \sum_{t=0}^{T-1} \mathbb{E}\left\{Q^c_{\text{NA}}(t)\right. \\ &\left.+Q^c_{\text{NNA}}(t) + Q^n(t)\right\} \leq \frac{(B + VP^*(\epsilon, \epsilon))T}{\epsilon}. \end{aligned} \quad (38)$$

By dividing both sides of (38) with T and using the fact that $L(t) \geq 0$ and taking as $T \rightarrow \infty$, this completes the proof of NA and NNA queue bounds (21).

Next, we prove average power bound (22). By applying $\epsilon(Q^c_{\text{NA}}(t) + Q^c_{\text{NNA}}(t) + Q^n(t)) \geq 0$ on (36), we obtain following inequality:

$$\begin{aligned}\Delta(L(t)) + V\mathbb{E}\{P^{c,\text{OPT}}(t) + P^{n,\text{OPT}}(t)\} \\ \leq B + VP^*(\epsilon, \epsilon) \end{aligned} \quad (39)$$

where $P^{c,\text{OPT}}(t)$ and $P^{n,\text{OPT}}(t)$ are CPU and network power consumption when the optimal algorithm (OPT)[21] is adopted. By summing (39) over $t = \{0, 1, \ldots T - 1\}$ and using the fact that $L(t) \geq 0$, $Q^c_{\text{NA}}(0) = 0, Q^c_{\text{NNA}}(0) = 0, Q^n(0) = 0$, and dividing T, we have

$$\frac{1}{T}\sum_{t=0}^{T-1} \mathbb{E}\{P^{n,\text{OPT}}(t) + P^{n,\text{OPT}}(t)\} \leq \frac{B}{V} + P^*(\epsilon, \epsilon). \quad (40)$$

[20]Then, we can use the law of iterated expectation.

[21]OPT is to exhaustively find $(\theta(t), s(t), l(t), i(t))$ which minimize the RHS of (10).

Then, taking as $T \rightarrow \infty$ and using the Lebesgue's dominated convergence theorem, this completes the proof of average power bound (22). ■

ACKNOWLEDGMENT

The authors would like to thank Hong-hwan Jeon in heaven for help with 3G/WiFi throughput measurement study.

REFERENCES

[1] J. Kwak, O. Choi, S. Chong, and P. Mohapatra, "Dynamic speed scaling for energy minimization in delay-tolerant smartphone applications," in *Proc. IEEE INFOCOM 2014*, Toronto, ON, Canada, 2014, pp. 2292–2300.

[2] J. Kang, S. Seo, and J. Hong, "Usage pattern analysis of smartphones," in *Proc. 13th Asia-Pacific Network Operations and Management Symp. (APNOMS)*, Taipei, Taiwan, Sep. 2011, pp. 1–8.

[3] J. Hwang, J. Lee, and N. Lee, "Change of media usage patterns due to the mobile Internet: Focusing on smartphone users," *Korea Information Society Development Institute (KISDI)*, vol. 1, pp. 1–210, Dec. 2010.

[4] A. Alexander, Smartphone Usage Statistics, 2012. [Online]. Available: http://ansonalex.com/infographics/

[5] Cisco Visual Networking Index: Global Mobile Data Traffic Forecast Update, 2015. Cisco, San Jose, CA, USA, 2015 [Online]. Available: http://www.cisco.com/c/en/us/solutions/collateral/service-provider/visual-networking-index-vni/white_paper_c11-520862.html

[6] Snapdragon S4 Processors: System on Chip Solutions for a New Mobile Age, White Paper, pp. 1–8, 2011.

[7] T. K. J. Chen, C. Yang, and C. Shih, "Energy-efficient real-time task scheduling in multiprocessor DVS systems," in *Proc. 2007 Asia and South Pacific Design Automation Conf.*, Yokohama, Japan, Jan. 2007, pp. 342–349.

[8] Y. Liang, P. Lai, and C. Chiou, "An energy conservation DVFS algorithm for the android operating system," *J. Convergence*, vol. 1, no. 1, pp. 93–100, Dec. 2010.

[9] A. Goldsmith, *Wireless Communications*. Cambridge, U.K.: Cambridge Univ. Press, 2005.

[10] S. Ha, S. Sen, C. Joe-Wong, Y. Im, and M. Chiang, "TUBE: Time-dependent pricing for mobile data," in *Proc. ACM SIGCOMM 2012*, Helsinki, Finland, Aug. 2012, pp. 247–258.

[11] Cloud Application, Dropbox. [Online]. Available: http://www.dropbox.com/

[12] Google Play [Online]. Available: https://play.google.com/store/apps/collection/topselling_free

[13] Operating Systems, iOS 6.0. [Online]. Available: http://www.apple.com/ios/

[14] Operating Systems, Android [Online]. Available: http://www.android.com/

[15] K. Son and B. Krishnamachari, "Speedbalance: Speed-scaling-aware optimal load balancing for green cellular networks," in *Proc. IEEE INFOCOM 2012*, Orlando, FL, USA, Mar. 2012, pp. 2816–2820.

[16] A. Wierman, L. L. Andrew, and A. Tang, "Power-aware speed scaling in processor sharing systems," in *Proc. IEEE INFOCOM 2009*, Rio de Janeiro, Brazil, Apr. 2009, pp. 2007–2015.

[17] M. Andrews, A. Anta, L. Zhang, and W. Zhao, "Routing for energy minimization in the speed scaling model," in *Proc. IEEE INFOCOM 2010*, San Diego, CA, USA, Mar. 2010, pp. 1–9.

[18] Kernel Governors, Modules, I/O Scheduler, CPU Tweaks AIO App Configs. [Online]. Available: http://forum.xda-developers.com/showthread.php?t=1369817

[19] A. Rahmati and L. Zhong, "Context-for-wireless: Context-sensitive energy-efficient wireless data transfer," in *Proc. ACM MobiSys*, San Juan, Puerto Rico, Jun. 2007, pp. 165–178.

[20] M. Ra, J. Peak, A. Sharma, R. Govindan, M. Krieger, and M. Neely, "Energy-delay tradeoffs in smartphone applications," in *Proc. ACM MobiSys*, San Francisco, CA, USA, Jun. 2010, pp. 255–270.

[21] P. Shu, F. Liu, H. Jin, M. Chen, F. Wen, Y. Qu, and B. Li, "Etime: Energy-efficient transmission between cloud and mobile devices," in *Proc. IEEE INFOCOM 2013*, Turin, Italy, Apr. 2013, pp. 14–19.

[22] K. Lee, J. Lee, Y. Yi, I. Rhee, and S. Chong, "Mobile data offloading: How much can wifi deliver?," *IEEE/ACM Trans. Netw.*, vol. 21, no. 2, pp. 536–550, Apr. 2013.

[23] L. Wang, W. Liu, A. Chen, and K. Yen, "Joint rate and power adaptation for wireless local area networks in generalized Nakagami fading channels," *IEEE Trans. Veh. Technol.*, vol. 58, no. 3, pp. 1375–1386, Mar. 2009.

[24] Y. Xu, J. Lui, and D. Chiu, "Improving energy efficiency via probabilistic rate combination in 802.11 multi-rate wireless networks," *Ad Hoc Networks*, vol. 7, no. 7, pp. 1370–1385, Feb. 2009.

[25] S. Wong, Y. Hao, L. Songwu, and B. Vaduvur, "Robust rate adaptation for 802.11 wireless networks," in *Proc. ACM MobiCom 2006*, Los Angeles, CA, USA, Sep. 2006, pp. 146–157.

[26] M. Ismail, W. Zhuang, and S. Elhedhli, "Energy and content aware multi-homing video transmission in heterogeneous networks," *IEEE Trans. Wireless Commun.*, vol. 12, no. 7, pp. 3600–3610, Jul. 2013.

[27] M. Ismail and W. Zhuang, "Mobile terminal energy management for sustainable multi-homing video transmission," *IEEE Trans. Wireless Commun.*, vol. 13, no. 8, pp. 4616–4627, Aug. 2014.

[28] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, pp. 1–211, 2010.

[29] Y. Yao, L. Huang, A. Sharma, L. Golubchik, and M. Neely, "Data centers power reduction: A two time scale approach for delay tolerant workloads," in *Proc. IEEE INFOCOM 2012*, Orlando, FL, USA, Mar. 2012, pp. 1431–1439.

[30] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," in *Proc. IEEE INFOCOM 2014*, Toronto, ON, Canada, Apr. 2014, pp. 1123–1131.

[31] M. Shin, S. Chong, and I. Rhee, "Dual-resource TCP/AQM for processing-constrained networks," *IEEE/ACM Trans. Netw.*, vol. 16, no. 2, pp. 435–449, Apr. 2008.

[32] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *Proc. 9th ACM SIGCOMM Conf., Internet Measurement Conf.*, Barcelona, Spain, Aug. 2009, pp. 280–293.

[33] Power Meter Device: Monsoon Power Monitor. [Online]. Available: http://www.msoon.com/LabEquipment/PowerMonitor/

[34] Advanced Task Killer Application. [Online]. Available: https://play.google.com/store/apps/details?id=com.rechild.advancedtaskkiller

[35] FFmpeg Media Encoder. [Online]. Available: https://play.google.com/store/apps/details?id=com.silentlexx.ffmpeggui&hl=ko

[36] Dataset for Statistics and Social Network of YouTube Videos. [Online]. Available: http://netsg.cs.sfu.ca/youtubedata/

[37] E. Perahia and S. Robert, *Next Generation Wireless LANs: 802.11n and 802.11ac*. Cambridge, U.K.: Cambridge Univ. Press, 2013.

[38] L. Kleinrock, *Queueing Systems*. New York, NY, USA: Wiley, 1975.

[39] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," in *Proc. IEEE INFOCOM 2014*, Toronto, ON, Canada, Apr. 2014, pp. 1123–1131.

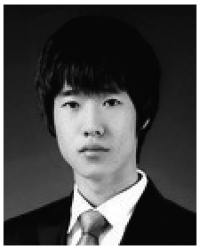
[40] NSTools Application and Open Source Code v1.16. [Online]. Available: http://forum.xda-developers.com/showthread.php?t=1333696

[41] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Found. Trends Netw.*, vol. 1, no. 1, pp. 1–149, 2006.

Jeongho Kwak received the B.S. degree (*summa cum laude*) in the Division of Electrical and Computer Engineering at Ajou University, Daejeon, Korea, in 2008, and the M.S. and Ph.D. degrees in the Department of Electrical Engineering at Korea Advanced Institute of Science and Technology (KAIST), Daejeon, Korea, in 2011 and 2015, respectively.

He is currently a Postdoctoral Research Associate with the Department of Electrical Engineering at KAIST. His research interests are in the areas of mobile cloud offloading systems, energy-efficient mobile systems, green cellular networks, and radio resource management in wireless networks.

Dr. Kwak received the Samsung HumanTech Thesis Prizes in 2013, 2014, and 2015 (Bronze, Silver, and Gold Prizes in Communication and Networks area, respectively), KAIST-LG Electronics 5G Best Paper Award in 2014 and Qualcomm Innovation Award in 2015.



Okyoung Choi received his B.S. and M.S. degrees in the Department of Electrical Engineering from Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2010 and 2012, respectively. He is currently a Ph.D. student in the Department of Electrical Engineering at KAIST. His research interests are in the areas of vehicular networks, delay-tolerant networks and energy efficiency of smart devices.

Song Chong (M'95) received the B.S. and M.S. degrees from Seoul National University and the Ph.D. degree from the University of Texas at Austin, all in electrical engineering.

He is a Professor in the Department of Electrical Engineering at Korea Advanced Institute of Science and Technology (KAIST) and was the Head of Communication and Computing Division of the department in 2009–2010. He is the Founding Director of KAIST-LGE 5G Mobile Communications & Networking Research Center funded by LG Electronics. Prior to joining KAIST in March 2000, he was with the Performance Analysis Department, AT&T Bell Laboratories, Holmdel, NJ, USA, as a Member of Technical Staff. His current research interests include wireless networks, mobile networks and systems, network data analytics, distributed algorithms, and cross-layer control and optimization.

Dr. Chong is currently an Editor of IEEE/ACM TRANSACTIONS ON NETWORKING, IEEE TRANSACTIONS ON MOBILE COMPUTING, and IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS. He is the Technical Program Committee Co-Chair of IEEE SECON 2015 and has served on the Technical Program Committee of a number of leading international conferences, including IEEE INFOCOM, ACM MobiCom, ACM CoNEXT, ACM MobiHoc, and IEEE ICNP. He serves on the Steering Committee of IEEE WiOpt and was the General Chair of IEEE WiOpt 2009. He received the IEEE William R. Bennett Prize in 2013, given to the best original paper published in IEEE/ACM TRANSACTIONS ON NETWORKING in 2011–2013, and the IEEE SECON Best Paper Award in 2013.

Prasant Mohapatra (M'89–SM'98–F'10) received the doctoral degree from Pennsylvania State University, State College, PA, USA, in 1993.

He is a Professor in the Department of Computer Science at the University of California, Davis. He was the Department Chair of Computer Science during 2007–2013, and held the Tim Bucher Family Endowed Chair Professorship during that period. In the past, he has been on the faculty at Iowa State University and Michigan State University. He has also held Visiting Scientist positions at Intel Corporation, Panasonic Technologies, Institute of Infocomm Research (I2R), Singapore, and National ICT Australia (NICTA). He has been a Visiting Professor at the University of Padova, Italy and Yonsei University, and KAIST, South Korea. His research interests are in the areas of wireless networks, mobile communications, cybersecurity, and Internet protocols. His research has been funded through grants from the National Science Foundation, US Department of Defense, Army Research Labs, Intel Corporation, Siemens, Panasonic Technologies, Hewlett Packard, Raytheon, Huawei Technologies, and EMC Corporation.

Dr. Mohapatra is the Editor-in-Chief of the IEEE TRANSACTIONS ON MOBILE COMPUTING. He has served on the editorial board of the IEEE TRANSACTIONS ON COMPUTERS, IEEE TRANSACTIONS ON MOBILE COMPUTING, IEEE TRANSACTION ON PARALLEL AND DISTRIBUTED SYSTEMS, ACM WINET, and *Ad Hoc Networks*. He has served as the Program Chair and the General Chair and has been on the program/organizational committees of several international conferences. He has been a Guest Editor for *IEEE Network*, IEEE TRANSACTIONS ON MOBILE COMPUTING, *IEEE Communications*, *IEEE Wireless Communications*, and *IEEE Computer*. He received an Outstanding Engineering Alumni Award in 2008 from Penn State. He also received an Outstanding Research Faculty Award from the College of Engineering at the University of California, Davis. He received the HP Labs Innovation awards in 2011, 2012, and 2013. He is a Fellow of the AAAS.