Performance-aware Energy Optimization on Mobile Devices in Cellular Network

Yong Cui*, Shihan Xiao*, Xin Wang†, Minming Li‡, Hongyi Wang* and Zeqi Lai*
*Tsinghua National Laboratory for Information Science and Technology
Dept. of Computer Science and Technology, Tsinghua University, Beijing, 100084, China
†Department of Electrical and Computer Engineering, Stony Brook University, Stony Brook, New York, USA
‡Department of Computer Science, City University of Hong Kong, Hong Kong, China
Email: cuiyong@tsinghua.edu.cn, xiaoshihan.xsh@gmail.com, xwang@ece.sunysb.edu, minming.li@cityu.edu.hk

***Abstract*—In cellular networks, it is important to conserve energy while at the same time ensuring users to have good transmission experiences. The energy cost can result from tail energy due to the radio resource control strategies designed in cellular networks and data transmission. Existing efforts generally consider one of the energy issues, and also ignore the adverse impact on user transmission performance due to energy conservation. In addition, many existing algorithms are based on prediction and knowledge on future traffic, which are hard to apply in a practical wireless system with dynamic user traffic and channel condition.**

The goal of this work is to design an efficient online scheduling algorithm to minimize energy consumption both due to tail energy and transmissions while meeting user performance expectation. We prove the problem to be NP-hard, and design a practical online scheduling algorithm *PerES* to minimize the total energy cost of multiple mobile applications subject to user performance constraints. We propose a comprehensive performance cost metric to capture the impacts due to task delay, deadline violation, different application profiles and user preferences. We prove that our proposed scheduling algorithm can make the energy consumption arbitrarily close to that of the optimal scheduling solution. The evaluation results demonstrate the effectiveness of our scheme and its higher performance than peers. Moreover, by supporting dynamic performance requirement by mobile users, *PerES* can achieve 2 times faster convergence to both the performance degradation bound and optimal energy conversation bound than those of traditional static methods. Using 821 million traffic flows collected from a commercial cellular carrier, we verify our scheme could achieve on average 32%-56% energy savings with different levels of user experience.

I. Introduction

There is a quick growth of cellular applications in recent years thanks to the constant increase of the processor power of mobile devices and the transmission bandwidth of cellular networks. The capacity of batteries, however, grows at much slower speed and the limited battery life has become the bottleneck of enhancing the user experience of mobile applications.

This work is supported by National Natural Science Foundation of China (no. 61120106008), National High Technology Development 863 Program of China (no. 2013AA010401). Xin Wang's research is supported by U.S. NSF grants CNS-127924 and ECCS-1231800. The work is also partly supported by a grant from the Research Grants Council of the Hong Kong Special Administrative Region, China (Project No. CityU 122512).

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

Energy conservation is often supported by existing wireless MAC protocols with different features. In UMTS (Universal Mobile Telecommunications System) network (one of the most popular 3G mobile communication technologies), the radio of a mobile device follows a *radio resource control* (RRC) mechanism, and the radio does not turn to a low power state immediately after data transmissions. Instead, a radio will stay at the high power state and wait for the expiration of an inactivity timer. If no transmission occurs during that period, it will then switch to the low power state. This period is defined as *tail time* and the corresponding energy consumption is called *tail energy*. The tail time is designed to avoid high signaling overhead of the 3G radio access network [1], and the tail time is also introduced in 4G LTE networks [2] recently. If a large amount of tail time is introduced, however, the energy-efficiency will deteriorate sharply.

Some recent efforts have been made to address this issue from different perspectives [3], [4]. Timer optimization is used in [5], [6] to adapt the RRC configurations in response to dynamic and complex traffic patterns. *Top* [7] utilizes the fast dormancy strategy and traffic predication to proactively demote the transmission to a low power state by optimizing the inactivity timer. To reduce the tail energy, application-layer solutions are proposed in [8]–[12] to schedule data transmissions. Work in *TailEnder* [8] proposes an online transmission scheduling algorithm to minimize the tail energy. Predication methods are applied in *Bartendr* [13] to vary transmission signal power to achieve higher energy efficiency in the mobile environment. However, the dynamics in mobile traffic and wireless link quality make conventional energy conservation algorithms based on predication difficult to apply. To overcome this limitation, recent works have introduced the *Lyapunov optimization framework* as a non-predication based online scheduling design principle [14], [15]. *SALSA* [14] took both the delay impact and wireless link quality into account and introduced the Lyapunov optimization framework as a general multi-interface online scheduler. However, the tail energy in cellular networks is not modeled. Authors in *eTime* [15] further proposed a similar scheduling method for 3G and WiFi network

interfaces with the tail energy embedded in its 3G power model. Some efforts have also been made in [14]–[16] to exploit the heterogeneity of WiFi and 3G network access to optimize the overall transmission energy. Complex computation tasks can be transmitted to the cloud for further energy efficiency [17]. However, the deadline issues and bandwidth competitions among multiple applications are not addressed in these studies.

Based on existing solutions, we identify three significant challenges in optimizing the energy in mobile cellular networks. The first issue is that there exists scheduling conflicts between the *tail energy* reduction and *data transmission energy* optimization. On the one hand, *tail energy* can be reduced if tasks can be queued and transmitted in batch. On the other hand, it would reduce the *data transmission energy* if tasks can be transmitted upon good channel condition. However, it may not be easy to find the optimal waiting time in order to meet both requirements and minimize the **hybrid energy**, i.e., the sum of the tail energy and data transmission energy. Previous works often focus on solving one of the problems without considering the other, while both are important and have big impact on the total system energy consumption.

As the second issue, although it is important to reduce the energy cost, it is equally important to ensure users to have good transmission experiences. The attempt to reduce both the tail and data transmission energy often introduce delay in packet transmissions. In addition, the user expectation on performance depends on many factors, such as the task types, application profiles, and user preferences. While targeting for energy optimization, existing works often ignore its impact on user transmission performance.

The third issue is the feasibility in real systems. For schemes that need lower-layer system support such as optimizing the timer of RRC, the configurations of RRC need to be modified which are usually decided by the cellular network carriers. Also, a transmission scheduling mechanism often needs to deal with transport layer (e.g., the congestion control and flow control of TCP) and application layer (e.g., interfaces exposed to applications) protocols and configurations.

Aiming at these challenges, the main contributions of our work are summarized as follows:

- We model the *hybrid energy optimization problem* in mobile cellular networks and prove its NP-Hardness.
- We design a practical online scheduler that can self-adapt to meet various user requirements on application performance. We prove both its energy conservation bound and performance degradation bound under varying mobile traffic and wireless link quality.
- We propose a novel method to enable a high convergence speed to both the optimal energy bound and required performance bound in a fast and practical way.
- Our scheme is implemented as a traffic management application on mobile devices, thus all the benefits can be achieved by solely upgrading the software.

The remainder of this paper is organized as follows. Section II introduces our basic problem formulation. In Section III, we provide scheduling analysis and our scheduling design. We evaluate the performance of our online scheduling algorithm in IV, and conclude our work in Section V.

II. Problem Formulation

In this section, we will elaborate on how we handle the performance and energy issues and formulate the transmission scheduling problem.

A. Basic Transmission Model

Our transmission framework runs as a daemon to collect the traffic generated by different applications, and the traffic is scheduled to transmit in the unit of time slot. The traffic from an application can be divided into multiple transmission tasks, where typically a task u corresponds to a packet. We denote the time slot that a task u arrives as $t_a(u)$, and the slot that u is scheduled to send to the corresponding socket as $t_s(u)$.

We embody the bandwidth of the cellular network with the capacity of a time slot, i.e., the maximum amount of data in bytes that can be transmitted between mobile devices and the base station in the slot. Let $c(t)$ denote the capacity of a slot t and $\upsilon_u(t)$ denote the data transfer rate of a transmission task u in the slot t. The condition

$$\sum_{u \in \{u | t_s(u) = t\}} \upsilon_u(t) \leq c(t) \tag{1}$$

should be met for any slot.

B. Performance Impact

In order to reduce the tail energy associated with the RRC process, an online scheduler can send the traffic in batch. However, as the mobile traffic and wireless link quality can not be accurately predicted, this process will inevitably introduce some delay in transmission and may even lead to violation of performance bounds such as the *deadline* of a task. To capture the performance impact, we introduce a performance cost metric $\phi_u(\cdot)$ by considering the views from three parties.

- **Task View** Different tasks generally have different delay tolerance. When the delay expectation for a task is violated, its performance may degrade significantly. This would cause bad user experience, and thus a large performance cost. We take the term *deadline* as the bound of the tolerable waiting delay of a task. In reality, iPhone users are allowed to specify their delay tolerance per application to improve their experiences.
- **Application View** Users have different delay experiences for different applications. We take the term *profile* to describe how the performance will decay with the increase of delay corresponding to an application. For example, the profile can be in linear degradation form with the linear increase of delay.
- **User View** Different mobile users may have different performance expectation on different types of applications. We apply *weight* to represent a user's preference on the performance of an application.

The *performance degradation function* ϕ_u can be computed as:

$$\phi_u(delay) = w_u \times f(delay) \times \mathcal{S}(u) \quad (2)$$

where $\mathcal{S}(u)$ denotes the data size in bytes of a task u. The *profile function* f represents the sensitivity of an application to the delay, and the *weight* w_u represents the user's preference on the application that generates u. Denote $t_d(u)$ as the deadline of a task u, we can easily get the following property:

Property 1. *Any $\phi_u(\cdot)$ should satisfy the following conditions:*

- $\phi_u(0) = 0$
- *If* $d_1 < d_2$, *then* $\phi_u(d_1) \leq \phi_u(d_2)$
- *If* $d_1 \leq t_d(u) - t_a(u) < d_2$, *then* $\phi_u(d_1) < \phi_u(d_2)$

The first two conditions ensure that the function ϕ_u captures the non-decreasing feature between the performance cost and task delay. The third condition reflects the cost associated with the violation of deadline, i.e., the user may have significantly worse experience thus higher performance cost.

Given $\phi_u(\cdot)$ for all the tasks in the pending transmission task set U, we can evaluate the total performance cost $\Phi(U, t_s(U))$ caused by the schedule $t_s(U)$ as $\sum_{u \in U} \phi_u(t_s(u) - t_a(u))$.

C. Energy Consumption for Data Transmission

For any given size of data units, the *data transmission energy* depends on the product of two factors: the transmission power and the time taken to transmit one bit of data. Previous works [13], [16], [18] have already illustrated how these factors vary with the signal strength. We take the RSSI (Received Signal Strength Indicator) value, i.e., the signal strength to evaluate the wireless link quality as it can be easily acquired on modern mobile devices without additional cost. We exploit the results from [13] and our measurements to estimate the data transmission energy. We define the signal strength as a time-varying function $signal(t)$. The function $R_{sig}(signal)$ maps a received signal strength to the data rate value, and the power to generate the signal is denoted by $P_{sig}(signal)$.

Let $t_r(u)$ denote the transmission time consumed by a transmission unit u and $t_e(u)$ denote the ending transmission slot that satisfies $t_e(u) = t_s(u) + t_r(u)$. Suppose two data units u_1 and u_2 that already arrived in the buffer are scheduled to transfer one after the other, the total data transmission time consumed is $t_r(u_1) + t_r(u_2)$ where each data unit takes the complete bandwidth resource for its transmission. Alternatively, the two can be transmitted concurrently with each using part of the bandwidth. In this case, the total data transmission time will remain the same. However, it would result in some performance loss since the completion time of one task gets longer. Therefore, to reduce the energy consumption and also to improve the transmission performance, it is more efficient to schedule all the data units sequentially. In other words, for any feasible solution $t_s(U)$, when we sort u in U in the ascending order by $t_s(u)$, we have $t_s(u_i) \geq t_e(u_{i-1}),\ 2 \leq i \leq n$.

Denote $\tau \in N^+$ as a discrete time slot and Δt_0 as the time length of one slot. Since the data of u will be transferred by consuming the data rate $c(\tau)$ in each slot τ from $t_s(u)$ to $t_e(u)$, we have the data size of the task u as

$$\mathcal{S}(u) = \sum_{\tau=t_s(u)}^{t_e(u)-1} c(\tau)\Delta t_0 = \sum_{\tau=t_s(u)}^{t_e(u)-1} R_{sig}(signal(\tau))\Delta t_0 \quad (3)$$

Let $E_{trans}(t_1, t_2)$ denote the *data transmission energy* consumed from the slot t_1 to t_2. If the data is transferred by consuming the transmission power $P_{sig}(signal(\tau))$ of the wireless interface in each slot τ from t_1 to t_2, then the data transmission energy can be computed as

$$E_{trans}(t_1, t_2) = \sum_{\tau=t_1}^{t_2-1} P_{sig}(signal(\tau))\Delta t_0 \quad (4)$$

During time period Γ, the total data transmission energy to transmit U by a schedule $t_s(U)$ can be estimated as

$$\widetilde{E}_d(U, t_s(U), \Gamma) = \sum_{u \in U} E_{trans}(t_s(u), t_e(u)) \quad (5)$$

D. Tail Energy Consumption Estimation

In a UMTS network, radio resources are managed through RRC and a state machine is maintained for the radio. The state machine has three basic states: IDLE, DCH and FACH. Their radio power is denoted as p_I, p_D and p_F respectively. The transition among different states is mainly determined by the data traffic conditions [5]. If the radio is at IDLE or FACH, the arrival of a data transmission unit will trigger it to promote to a higher power state DCH (the transitions IDLE $\rightarrow$ DCH and FACH $\rightarrow$ DCH are called PROM1 and PROM2 respectively, see Fig. 1). If there are no transmission tasks arriving and the radio remains inactive for a time duration, it will lead to an instant state demotion, either DCH $\rightarrow$ FACH or FACH $\rightarrow$ IDLE. The *tail energy* wasted during both the time durations (denoted by δ_D and δ_F respectively) can significantly impact the total energy consumption.

For accurate estimation of energy consumption, it is necessary to measure the above-mentioned parameters. We follow the measurement schemes proposed in [5] to find these parameters. The measurement is conducted in a UMTS network in China (CHN-CUGSM) and we use a smartphone (Google

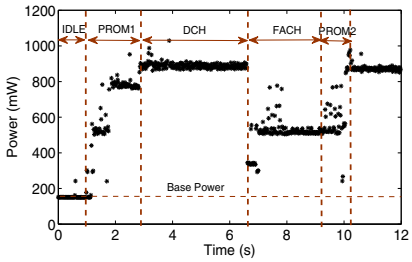


Fig. 1. Device Power at Different RRC States

TABLE I. Parameters of Different RRC States

	DCH	FACH	PROM1	PROM2
p(mW)	732.826	388.880	557.708	423.625
δ(s)	3.287	4.024	2.114	1.039

Nexus S) as the test platform. Fig. 1 is an example of the power at different states and Table I lists the detailed measurement results. The parameters recorded in Fig. 1 correspond to the total device power and we take power at IDLE state as the base to get the radio power.

Let T denote one complete tail time, i.e, the sum of δ_D and δ_F. Let the set of transmission tasks $U = \{u_i | 1 \leq i \leq n\}$ be sorted in the ascending order by $t_s(u)$ of each task u. The total tail energy consumption during the transmission of U according to a schedule $t_s(U)$ within a given period Γ can be estimated in a fine-grained way as

$$\widetilde{E}_t(U, t_s(U), \Gamma) = \sum_{2 \leq i \leq n} E_{tail}(\Delta t_i) + E_{tail}(T) \quad (6)$$

where $\Delta t_i = t_s(u_i) - t_e(u_{i-1})$ and

$$E_{tail}(t) = \begin{cases} p_D \cdot t & \text{if } 0 \leq t \leq \delta_D \\ p_D \cdot \delta_D + p_F(t - \delta_D) & \text{if } \delta_D < t \leq \delta_D + \delta_F \\ p_D \cdot \delta_D + p_F \cdot \delta_F & \text{otherwise.} \end{cases} \quad (7)$$

E. Optimization Problem

Our objective is to find a schedule $t_s(U)$ that can minimize the total energy consumption for transmitting data in U with the total performance degradation constrained below an upper-bound (denoted by $\tilde{\Phi}$) during a given time period Γ. That is

$$\sum_{u \in U} \phi_u(t_s(u) - t_a(u)) \leq \tilde{\Phi}. \quad (8)$$

A higher performance bound suggests a longer tolerable delay, which would provide more opportunities to transmit traffic in batch to reduce the tail energy cost. Based on (5) and (6), the total energy consumption can be calculated as $E(U, t_s(U), \Gamma) = \widetilde{E}_d(U, t_s(U), \Gamma) + \widetilde{E}_t(U, t_s(U), \Gamma)$.

Then the optimization problem is formulated as

$$\min E(U, t_s(U), \Gamma) \quad (9)$$
$$\text{subject to Constraints (1), and (8).}$$

Theorem 1. *Computing an optimal solution for the hybrid energy optimization problem in (9) is* NP*-hard.*

By constructing a specific problem instance, we can transform the NP-hard *partition problem* [19] into our problem. See the detailed proof in our technical report [20].

III. Scheduling Analysis and Design

In this section, we will introduce our scheduling analysis and solution design on the hybrid energy optimization problem.

A. Performance-aware Energy Scheduling Design

As finding the optimal solution of the hybrid energy problem is NP-hard in Theorem 1, it is difficult to be applied in the practical environment. Instead, we design an online scheduler, called **PerES** (Performance-aware Energy Scheduler), which can be easily run in the practical system. It will take into account the user requirements on performance for different applications and ensure stable system performance under dynamic user traffic and variation of wireless link quality.

1) Online Optimization: In our scheduling design, the online scheduler requires no future information of the traffic. It makes the scheduling decision in each time slot to obtain the long-term benefits, i.e., optimizing both the energy and performance in a long enough time scale ($\Gamma \to \infty$). We derive the practical optimization scheme as follows.

Define $PW(\tau)$ and $PD(\tau)$ as the energy consumed and performance cost in the time slot τ respectively. Let $\overline{PW}(t)$ and $\overline{PD}(t)$ define the time-average power consumption and the time-average performance degradation on user experience within a time period t respectively, we have

$$\overline{PW}(\Gamma) = \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} PW(\tau) = \frac{1}{\Gamma} E(U, t_s(U), \Gamma) \quad (10)$$

$$\overline{PD}(\Gamma) = \frac{1}{\Gamma} \sum_{\tau=0}^{\Gamma-1} PD(\tau) = \frac{1}{\Gamma} \sum_{u \in U} \phi_u(t_s(u) - t_a(u)) \quad (11)$$

where Γ is the scheduling period.

Define Ω as the user requirement on the maximum tolerable time-average performance cost of all applications. Then the following performance condition should be satisfied:

$$\overline{PD}(\Gamma) \leq \Omega \quad (12)$$

Based on (9), replace the objective $E(U, t_s(U), \Gamma)$ with $\overline{PW}(\Gamma)$, and the performance condition (8) with (12), the online optimization could finally be formulated as:

When $\Gamma \to \infty$,

$$\min \quad \overline{PW}(\Gamma) \quad (13)$$
$$\text{subject to Constraints (1) and (12).}$$

2) Group classify: To take into account the performance metric in our scheduling, we first classify applications into n groups $G = \{G_1, G_2, ..., G_n\}$. The **group** is classified by two properties: the form of *profile function* (decided by application profiles) and the performance *weight* (decided by user preference). If two tasks share the same properties, then they belong to the same group. Generally, the same type of applications are classified into one group.

3) Delay-aware queue management: Generally, the *delay sensitivity* of a task is reflected by its *deadline*. A smaller deadline often implies an application has higher delay sensitivity. Tasks with the same deadline can be put into one group and inserted into one of m queues based on their current delay. A queue q_{ij} corresponds to one **delay-level** j from group i. The delay-level interval is defined as d_i/θ where d_i denotes the *deadline* of tasks from group i, and θ is the delay granularity of dividing the group tasks into queues and also the granularity considered by the scheduler. Note that m is large enough so that the $m \times d_i/\theta$ is the maximum waiting delay that users can tolerate for any task in the buffer. If the delay value of a task u from group i is between $d_i/\theta \times (j-1)$ and $d_i/\theta \times j$, it will be put in q_{ij}; when its delay increases to exceed $d_i/\theta \times j$, it will be popped out of q_{ij} and be pushed into the next delay-level queue $q_{i(j+1)}$. The scheduler will take the delay level as

Algorithm 1 RAA: Rate Allocation Algorithm

Input: V, t // the current time slot
Output: $\mathbb{R}_m$ // the data rate allocation set
1: Update the delay-level queues by (15)
2: **for** each queue q_{ij} **do**
3: $r_{ij} \leftarrow 0$
4: compute A_{ij} by (19)
5: **end for**
6: $R_t \leftarrow R_{sig}(t)$
7: Sort the q_{ij} in the descending order of A_{ij} as Q
8: **while** $R_t > 0$ and $Q \neq \phi$ **do**
9: Pop the q_{ij} in Q who has the largest A_{ij}
10: $r_{ij} \leftarrow \min\{Size(q_{ij}), R_t\}$
11: $R_t \leftarrow R_t - r_{ij}$
12: **end while**
13: Get the rate allocation set $\mathbb{R}_m = \{r_{ij}\}$ and compute the objective D_m with V by (18)
14: **if** $D_m \leq 0$ **then**
15: Set all the elements in $\mathbb{R}_m$ to 0
16: **end if**
17: **return** $\mathbb{R}_m$

the reference to determine the transmission sequence and time, i.e., the delay of tasks in q_{ij} is taken as $d_i/\theta \times j$. Define the length of the minimum time unit as ℓ_t. If the delay-level interval d_i/θ equals ℓ_t, then $d_i/\theta \times j$ is equivalent to the accurate task delay. Let $F(\mathcal{D}(q_{ij}))$ denote the performance cost per byte of queue q_{ij}, i.e., $F(\mathcal{D}(q_{ij}))) = \phi_u(\mathcal{D}(q_{ij}))/Size(q_{ij})$ where $\mathcal{D}(q_{ij}) = d_i/\theta \times j$ is the task delay in q_{ij} and $Size(q_{ij})$ is the total data size of tasks in q_{ij}.

After the division of delay-level queues, the scheduler needs to decide the transmission rate for each queue in each time slot based on the current information. Denote $\gamma_{ij}(t)$ as the transmission task arriving rate for queue q_{ij}, and $r_{ij}(t)$ as the transmission rate allocated for queue q_{ij} at time slot t. When the application tasks arrive at the scheduler's buffer, their delay is zero and thus the tasks from applications will arrive only at the first delay-level queue q_{i1}. Therefore, $\gamma_{ij}(t) = 0$ when $j > 1$. Denote $PD_{ij}(t)$ as the total performance cost of the tasks in queue q_{ij} in time slot t, we have

$$\begin{aligned} PD_{ij}(t+1) = PD_{ij}(t) - \varphi_{ij} \times (r_{ij}(t) + \mathcal{O}_{ij}(t)) \\ + \varphi_{ij} \times (\gamma_{ij}(t) + \mathcal{I}_{ij}(t)) \end{aligned} \quad (14)$$

where $\varphi_{ij} \triangleq F(d_i/\theta \times j)$, while $\mathcal{O}_{ij}(t)$ and $\mathcal{I}_{ij}(t)$ denote the size of the tasks that are popped out of and pushed into the current queue q_{ij} at time slot t respectively. Note that $\sum_{j=1}^{m}\{\mathcal{I}_{ij}(t)\} = \sum_{j=1}^{m}\{\mathcal{O}_{ij}(t)\}$. By summing up the index j in (14), we have

$$PD_i(t+1) = PD_i(t) - \sum_{j=1}^{m} \varphi_{ij} r_{ij}(t) + \varphi_{i1}\gamma_i(t) \quad (15)$$

where $PD_i(t) \triangleq \sum_{j=1}^{m} PD_{ij}(t)$ and $\gamma_i(t) \triangleq \sum_{j=1}^{m} \gamma_{ij}(t)$.

4) Rate Allocation: In this part, we describe our *PerES*'s rate allocation algorithm (called **RAA**). Following the Lyapunov framework [21], our Lyapunov function is defined as $L(t) \triangleq \frac{1}{2}\sum_{i=1}^{n}(PD_i(t))^2$. Denote the vector $\overrightarrow{PD}(t) \triangleq \{PD_{ij}(t)|_{1\leq i\leq n, 1\leq j\leq m}\}$ as the performance cost of each queue at time t. Then the one-step Lyapunov drift $\Delta(t)$ is defined as:

$$\Delta(t) \triangleq \mathbb{E}\{L(t+1) - L(t)|\overrightarrow{PD}(t)\} \quad (16)$$

We add the energy minimization objective into the Lyapunov drift by the *drift-plus-penalty* form $\Delta(t) + V\mathbb{E}\{PW(t)|\overrightarrow{PD}(t)\}$ and generate the following lemma:

Lemma 1. *Assume that the data arrival process $\widetilde{\lambda}(t)$, and the transmission process $\widetilde{u}(t)$ have finite expectation, i.e., $\exists$ constants $\mathcal{A}$ and $\mathcal{U}$ such that $\mathbb{E}\{\widetilde{\lambda}(t)\} \leq \mathcal{A}$ and $\mathbb{E}\{\widetilde{u}(t)\} \leq \mathcal{U}$. We have*

$$\begin{aligned} &\Delta(t) + V\mathbb{E}\{PW(t)|\overrightarrow{PD}(t)\} \\ &\leq B - \mathbb{E}\{\sum_{i=1}^{n}[PD_i(t)\mathbb{E}\{\sum_{j=1}^{m}\varphi_{ij}r_{ij}(t)|\overrightarrow{PD}(t)\}] \\ &\quad - V \cdot PW(t)|\overrightarrow{PD}(t)\} + \sum_{i=1}^{n}[PD_i(t)\cdot\varphi_{i1}\gamma_i] \end{aligned} \quad (17)$$

where $B \triangleq \frac{1}{2}\{(\sum_{i=1}^{n}\sum_{j=1}^{m}\varphi_{ij}^2)\cdot\mathcal{U}^2 + (\sum_{i=1}^{n}\varphi_{i1}^2)\cdot\mathcal{A}^2\}$ and φ_{ij} is a positive constant.

By applying (15) into the Lyapunov drift (16), we can obtain the fact of (17). See the detailed proof in [20]. Based on the Lyapunov design principle, the optimal scheduling decision is to minimize the *drift-plus-penalty* expression in each time slot. *PerES* minimizes the RHS of (17) to guarantee the performance stability with the minimal power consumption:

$$\begin{aligned} Max\ D(t) = &\sum_{i=1}^{n}[PD_i(t) \times \sum_{j=1}^{m}\{\varphi_{ij} \times r_{ij}(t)\}] \\ &- V \times PW(t) \\ s.t. \quad &\sum_{i=1}^{n}\sum_{j=1}^{m} r_{ij}(t) \leq c(t) \\ &0 \leq r_{ij}(t) \leq Size(q_{ij}) \end{aligned} \quad (18)$$

where $c(t)$ is the bandwidth of the wireless link at time t. Note that the rate allocation constraints embed the bandwidth competition among different applications.

As Algorithm 1 illustrates, *RAA* first updates the performance degradation $PD_i(t)$ of all the queues by (15) and initializes the initial rate allocation set to all zeros. By solving the linear programming problem (18), *RAA* achieves the optimal rate allocation set $\mathbb{R}_m = \{r_{ij}|i = 1, 2, 3, ..., n; j = 1, 2, 3, ..., m.\}$ with the following procedure. First, compute the weight A_{ij} of each queue:

$$A_{ij} = PD_i(t) \times \varphi_{ij} = PD_i(t) \times F(d_i/\theta \times j) \quad (19)$$

Next, the total bandwidth $c(t) = R_{sig}(t)$ is allocated to the queues in the descending order of A_{ij} until it is completely allocated as lines 6-12 in Algorithm 1 show.

After rate allocation for delay-level queues, *RAA* computes the objective function value D_m by applying the rate allocation set $\mathbb{R}_m$ into the objective function of (18). For a quick

Algorithm 2 PerES: Performance-aware Energy Scheduler

Input: Ω, t //the current time slot
Output: Real-time Scheduling Decision
// Set the control parameter V using SVA (line 1-10)
1: **if** t equals 0 **then**
2: $V(t) = 0$
3: **else**
4: calculate $\overline{PD}(t)$ by (11)
5: **if** $\overline{PD}(t) < \Omega$ **then**
6: $V(t) = V(t-1) + \delta$
7: **else**
8: $V(t) = V(t-1)/2$
9: **end if**
10: **end if**
// Allocate rate for queues using RAA
11: Call *RAA*$(V(t), t)$ (Algorithm 1) to get $\mathbb{R}_m$
12: Allocate data rate to each queue q_{ij} by $\mathbb{R}_m$ for data transferring

computation for this control decision, $PW(t)$ can be computed as the sum of *data transmission energy* and *tail energy*:

$$PW(t) = P_{sig}(t) \times \{(\sum_{i=1}^{n}\sum_{j=1}^{m} r_{ij}(t))/c(t)\} + E_{Tail}(\Delta t') \quad (20)$$

where $\Delta t' = t - t_{last}$ and t_{last} denotes the last time slot that transfers data. If $D_m \leq 0$, then the rate allocation set $\mathbb{R}_m$ is set to all zeros.

As Algorithm 2 shows, *PerES* makes the scheduling decision in each slot. It first determines the control parameter V, and then calls *RAA* to set the rate allocation $\mathbb{R}_m$. *PerES* serves all the queues by $\mathbb{R}_m$ for data transferring. If all the elements in $\mathbb{R}_m$ equal zero, then *PerES* keeps all the queues waiting.

B. Bound and Convergence Analysis of PerES

Theorem 2. *Assume that the data arrival rate is strictly within the network capacity region, and the online scheduling decision (18) is applied by $PerES$ at each time slot. For any control parameter $V > 0$, the time-average power consumption $\overline{PW}_\infty$ and time-average performance cost $\overline{PD}_\infty$ satisfy that:*

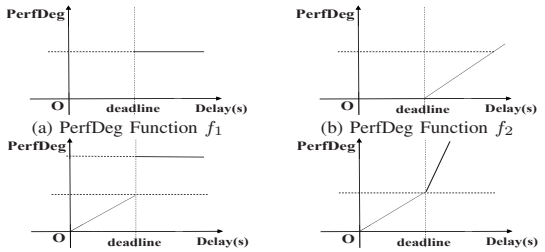
$$\overline{PW}_\infty = \lim_{\Gamma\to\infty} \sup \frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1} \mathbb{E}\{PW(\tau)\} \leq P^* + \frac{B}{V} \quad (21)$$

$$\overline{PD}_\infty = \lim_{\Gamma\to\infty} \sup \frac{1}{\Gamma}\sum_{\tau=0}^{\Gamma-1} \mathbb{E}\{PD(\tau)\} \leq \frac{B + VP^*}{\varepsilon} \quad (22)$$

where B and ε are positive constants. P^* is the theoretical optimal time-average power consumption.

Based on Lemma 1 and the similar method for deriving Lyapunov bounds in [14], [21], we could obtain the fact of (21) and (22). See the detailed proof in our technical report [20].

Inspired by Theorem 2, we could solve the online optimization problem (13) in the following way. For any given user requirement Ω, if we could find a proper V value that makes the system work with a time-average performance cost close to but within Ω, then according to (22), V is maximized to meet the performance constraint, while according to (21), the upper bound of the time-average power consumption is minimized to its optimal value. The traditional way of solving this convergency problem falls into the region of calculating a *magic number* of V based on some heuristic information, such as the method proposed in [14], or assuming a *nice range* of V that is possible to apply [15]. However, it is difficult to analyze the performance and hard to apply these conventional schemes in a practical and complex mobile environment.



(a) PerfDeg Function f_1 (b) PerfDeg Function f_2
(c) PerfDeg Function f_3 (d) PerfDeg Function f_4
Fig. 2. Different Functions of Performance Degradation

We design a dynamic scheme, called **SVA** (Self-adaptive V Algorithm) to handle this tough issue in a practical and fast way. As the lines 1-10 in Algorithm 2 show, the *SVA* monitors the current time-average performance cost value $\overline{PD}$. When it is lower than the user requirement, then *SVA* increases the V linearly; otherwise, *SVA* cuts the V value down to a half. The intuition behind *SVA* comes from the congestion avoidance scheme utilized by the TCP protocol. The AIMD (Additive Increase Multiplicative Decrease) is primarily designed for the TCP window to converge to a value that gets the system work close to but within the congestion bound, thus the throughput is maximized. Our experiment validates that using the *AIMD* scheme in our system could also enable a fast convergency to the required performance bound, and thus the time-average power consumption is minimized to its optimal bound.

IV. Evaluations

A. Evaluation Setup and Methodologies

We implement *PerES* as a traffic management application by utilizing IPTABLES (a system tool in Android) to redirect and buffer the transmission tasks generated from other network applications on a smartphone (Google Nexus S). We monitor the signal strength of the cellular network interface and then measure the energy consumption and performance metrics. To acquire the transmission rate and power under different signal strength, we take the phone to record 3G signal traces in 20 different places of Tsinghua University. The power value is measured by the Monsoon Power Monitor device. We find that the power P_{sig} (mW) and the data rate R_{sig} (kBps) could be fitted as a linear function with the signal strength [13], [16]: $P_{sig}(t) = -25 * Signal(t) - 1030.9$ and $R_{sig}(t) = 2.667 * Signal(t) + 293.73$. We evaluate the performance of scheduling algorithms through simulations over a period Γ of 10000 time slots (one second as one slot). The signal is a sine function in the range of -50dBm to -110dBm with a random interference between -10dBm and 10dBm, and its period is set to 500 time slots. The task arrives following

a *Poisson Distribution*, and the average arrival interval of the tasks is set as their waiting *deadline* to capture the features of applications that have periodic updates. The data size of one task is set as a random variable in $(0, 500]$ kBytes. The linear coefficient δ of *SVA* is set to 0.001. The detailed application settings are shown in Table II.

We analyze the impact of various parameters listed in Table IV. During the simulation, when one factor is changed, other factors are set to their default values. We evaluate four representative profile functions in Fig. 2. All functions satisfy Property 1, but have different forms before and after the deadline. Function f_1 captures the features of the multimedia streaming applications that can prefetch and buffer for some transmission frames, thus users will not experience the performance loss caused by the task delay until certain deadline is violated. Function f_2 captures the features of the email applications that will not trigger the users' concerns until certain user tolerance time is violated, and then cause worse experience if further delay is introduced. Function f_3 captures the features of the SNS applications that have periodic updates of servers' information, thus will cause increasing performance cost on experience with delay. When the deadline is violated, it keeps as a constant value because a user will receive the next query task to get the updates. Function f_4 captures the features of the real-time applications like network games that will cause worse user experience with increased information delay. Each data point recorded in our simulation results is the average value over 20 random problem instances.

For comparative analysis, four non-predication based online schedulers are evaluated, i.e., *TailEnder* [8], *SALSA* [14], *eTime* [15] and our *PerES*. TailEnder is a const-setting-based online scheduler while the other three are designed under the Lyapunov optimization framework. Table III shows their different features. They represent the consideration for signal strength impact, different granularity of tail energy counting, deadline awareness, application profiles, user preference and convergence scheme separately. For each problem instance solved by *eTime* and *SALSA*, we set their V parameter (the trade-off parameter in Lyapunov framework) following their rules introduced in [14], [15]. Their multi-interface selection is limited to the cellular network interface only. Moreover, we develop the *E-P panel* to compare the scheduling optimality for the total energy consumption *E* and performance degradation *P* ($\sum_{\tau=0}^{\Gamma-1} PD(\tau)$) under different settings. In the E-P panel, each set of points for schedulers is acquired by linearly increasing the parameter V of *SALSA*, *eTime* and the Ω of *PerES* in equal pace, and there is no parameter change in *TailEnder*. We fit a trend curve for each set of points acquired by one scheduler.

TABLE II. Application Settings

App ID	1	2	3	4	5
Deadline (s)	10	200	400	800	1600
Weight	1/10	1/200	1/400	1/800	1/1600

TABLE III. Evaluation Schedulers

Scheduler	Signal	Tail	Deadline	Profile	Weight	Convergence
TailEnder	-	Coarse	✓	-	-	-
SALSA	✓	-	-	-	-	Static
eTime	✓	Coarse	-	-	-	Static
PerES	✓	Fine	✓	✓	✓	Dynamic

TABLE IV. Evaluation Parameters Setup

	Default	Range
Minimum Arrival Interval (s)	10	$1 \sim 100$
Minimum Deadline (s)	10	$1 \sim 100$
Maximum Preference Weight	1/10	$1/100 \sim 1$
Average Signal Strength (dBm)	-80	$-110 \sim -50$
Profile Function	f_4	$\{f_1, f_2, f_3, f_4\}$
D-L Accuracy (θ)	10	$1 \sim 100$

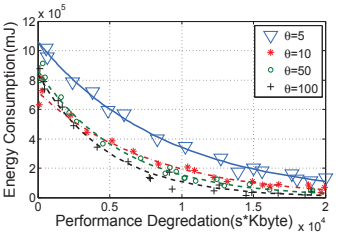


Fig. 3. The E-P Panel of Delay-granularity Impact

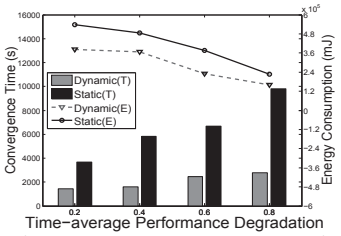


Fig. 4. Convergence: Dynamic vs. Static Strategy

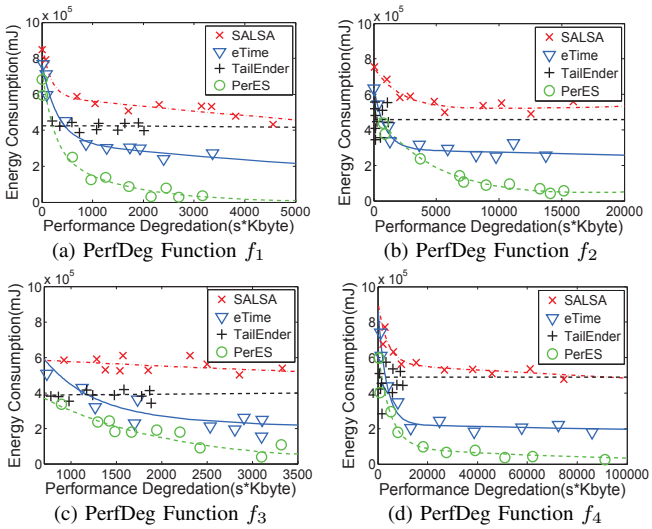
We evaluate the tail energy and data transmission energy for energy metrics, and the deadline-violation ratio, normalized average delay and average throughput for performance metrics. The *deadline violation ratio* denotes the total size of tasks whose delay exceeds their deadline divided by the total data size. The *normalized average delay* $\overline{D}$ denotes the sum of weighted (normalized preference weight) delay of all tasks divided by the total data size of all tasks: $\overline{D} = \frac{\sum_u \{w_u \times Delay_u \times Size_u\}}{\sum_u \{Size_u\}}$. The *average throughput* denotes the size of data transferred over the given simulation period.

B. Parameters Analysis

1) Impact of the Delay-granularity Setup: As Fig. 3 shows, we test four levels of accuracy setting for the *E-P panel*. As θ increases, the delay granularity for scheduling decreases. We can find that, from $\theta = 5$ to 10, the performance has a big improvement, and the energy consumption decreases faster with the performance degradation for $\theta = 10$. When the granularity further decreases, i.e., $\theta = \{10, 50, 100\}$, and performance does not have significant change. In the following, we set $\theta = 10$ to achieve the best balance between the overhead of queue management and the good performance of scheduling.

2) Dynamic vs. Static Schemes: As Fig. 4 shows, we set four levels of time-average performance requirement Ω between 0.2 and 0.8. In our study, for a given Ω setting, the static strategy uses the optimal static V value obtained by adjusting the V value in the experiment to make the time-average performance degradation converge to Ω exactly, and keeps the V value as a constant during the scheduling. We find that *SVA* (dynamic strategy) converges to the performance target 2-5 times faster than that of the static strategy. We can see that larger performance requirement Ω implies larger room for energy optimization. Furthermore, for the same performance requirement, a faster convergence speed of *SVA* results in higher energy efficiency than the static strategy during a specified scheduling period. Therefore, our dynamic scheme can react fast to the change of user requirements.

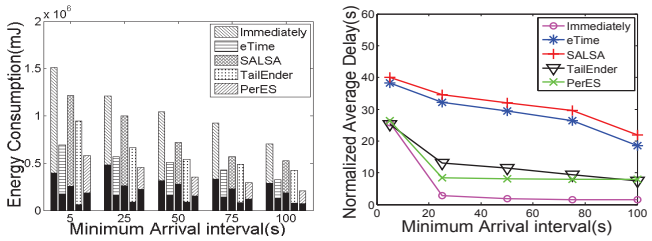
3) Profile Functions: As Fig. 5 shows, we evaluate schedulers in *E-P panels* with different profile functions listed in



(a) PerfDeg Function f_1 (b) PerfDeg Function f_2

(c) PerfDeg Function f_3 (d) PerfDeg Function f_4

Fig. 5. The E-P Panels for Different Profile Functions



(a) Energy Consumption (b) Normalized Average Delay

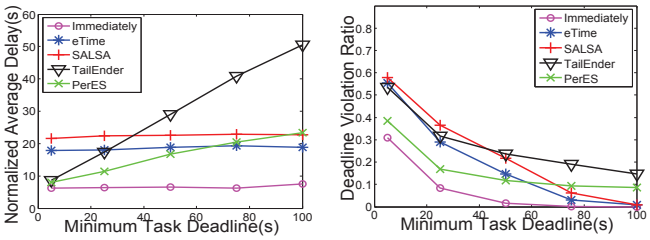
Fig. 6. Performance over Different Arrival Interval

Fig. 2. We could observe the obvious tradeoffs between the energy consumption and performance degradation for all other schemes except TailEnder, where the faster dropping of energy with the increasing degradation of performance indicates a better performance. *eTime* performs better than *SALSA* by embedding the tail energy into its control decision. Our *PerES* performs the best for all the profile functions tested. On average its energy efficiency is 2 times that of the *eTime* and *TailEnder*, and 4 times that of the *SALSA* for a given performance. TailEnder's results are constrained within a small range, as it schedules the transmission mostly around the deadline. This helps to improve its performance but limits the room for its energy optimization.

C. Comparative Analysis

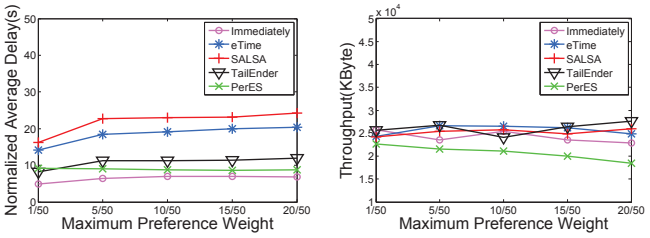
We compare the performance of schedulers in Table III. The default scheduling scheme on mobile devices, called *Immediately*, is also included as a reference, i.e., it intends to transfer the data immediately upon the arrival of tasks.

1) Arrival Pattern Impact: In Fig. 6(a), the black part of the energy metric represents the tail energy part while the non-black part represents the data transmission energy part. As expected, heavier traffic results in larger energy and performance cost. However, the distributions of the tail energy and data transmission energy have big difference among different schedulers. The tail energy of *SALSA* is on average twice that of *eTime*, *PerES* and *TailEnder*. This is because *SALSA*



(a) Normalized Average Delay (b) Deadline Violation Ratio

Fig. 7. Performance over Different Task Deadline



(a) Normalized Average Delay (b) Average Throughput

Fig. 8. Performance over Different User Preference Weight

does not consider the tail energy in its decision. Further, because *TailEnder* is not aware of the signal variation, its data transmission energy is on average twice that of other schedulers. *PerES*'s energy efficiency is similar to *eTime* and performs the best under different traffic conditions. In Fig. 6(b), since both *PerES* and *TailEnder* are aware of the task deadline, they have less normalized average delay than *eTime* and *SALSA*.

2) Task Deadline Impact: In Fig. 7(a), when the deadline is small, both *TailEnder* and *PerES* achieve smaller average delay. However, when the deadline is large enough, they will generate larger average delay than *SALSA* and *eTime*. As *SALSA* and *eTime* are not deadline-aware, they generate a stable average delay. However, *TailEnder* and *PerES* intend to transfer tasks closer to their deadline to increase energy efficiency, thus have higher delay when the deadline increases. The delay of *PerES* increases much slower than that of *TailEnder*. In Fig. 7(b), the deadline violation ratio of four schedulers decreases with increasing task deadline as expected. *PerES* achieves the lowest deadline violation ratio when the deadline is small and keeps a stable output when it turns large.

3) Weight Impact: The normalized average delay represents the delay from the user preference view. In Fig. 8(a), the normalized average delay of *SALSA*, *eTime* and *TailEnder* increases with the maximum preference weight while *PerES* keeps a low and stable value. This is because they do not consider the user preference in different tasks. In Fig. 8(b), the throughput of *PerES* is also the lowest and decreases when the preference weight increases. A high preference weight indicates that user cares more about the application performance (delay in the case evaluated) and the applications with higher weight will be given higher priority for transmissions. This could optimize the user experience on their preferred applications at the cost of the overall throughput reduction and additional delay of other applications. More extensive comparative results on other factors can be found in our technical report [20].

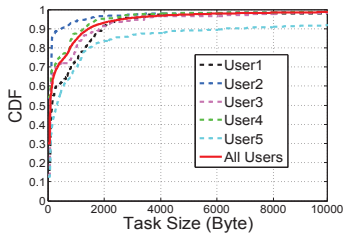


Fig. 9. The CDF of task size in real traffic scenarios

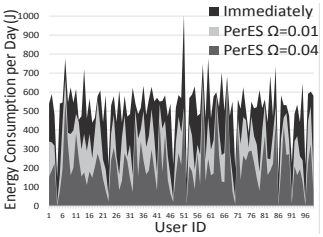


Fig. 10. Energy comparison for 100 users

D. Real-Traffic Application

We analyzed a large traffic flow trace from 99 collection points by a 3G UMTS carrier in China on January 10, 2013. The trace data capture about 821 million flow records (1.2 Terabytes). Each record corresponds to the information of one flow which contains the user IP, server IP, flow time stamps, uploading and downloading data size but without any user data. Normally, the same user IP corresponds to one specific user within some time window. To simplify the process, we set the time window as one day. We consider the uploading data in one flow as one task generated from network applications on mobile devices and the same server IP as one specific application.

As Fig. 9 shows, we first analyze the task size distribution in the traffic trace data. We randomly choose one collection point and pick the top 5 users that have the largest number of flows in one day for user-specific distribution. Then we derive the flows of all users collected by all collection points in one day for a general distribution. We can find that, most users have the similar task size distribution, and small-size data account for a major portion of the trace, i.e., above 90% of tasks have their data size smaller than 6 kBytes. This gives us the insight that most users have frequent task arrivals with a small data size, which will lead to a large fraction of tail energy.

As Fig. 10 shows, we randomly select 20 collection points and pick the top 5 users who own the largest number of flows in each collection point in one day. For each user, we select the top 5 applications that have the largest number of flows communicated with the user, and randomly select the signal trace in one day from the traces collected by real mobile users for 10 days. For the task flows of each user, we run the default scheme *Immediately* and our *PerES* implemented with different levels of performance requirement Ω on the phone for 100000 time slots respectively. On average for each user, in the case $\Omega = 0.01$, *PerES* achieves totally 32% energy savings (tail energy reduction by 32.5%) with the normalized average delay as 20s and the deadline violation ratio as 0.22; in the other case $\Omega = 0.04$, *PerES* achieves totally 56% energy savings (tail energy reduction by 60.7%) with the normalized average delay as 57s and the deadline violation ratio as 0.31.

V. Conclusion and Future Work

In this paper, we propose an adaptive online scheduling algorithm to improve the energy efficiency of mobile devices in cellular networks while also considering user performance need. Different from existing work, we formulate the application performance degradation problem on user experience and build a comprehensive metric to capture the impact of different performance factors and user preference. Evaluation results demonstrate the effectiveness of our proposed schemes in achieving higher performance over peer schemes. We further validate the high energy efficiency of our proposed scheduling algorithm under different user experiences through trace data collected. In the future work, we will investigate other methods for the global-optimal solution as a reference.

References

[1] 3GPP, "System impact of poor proprietary fast dormancy," *3GPP discussion and decision notes RP-090941*, 2009.
[2] J. Huang, F. Qian, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *MobiSys 2012*.
[3] Y. Cui, X. Ma, H. Wang, I. Stojmenovic, and J. Liu, "A survey of energy efficient wireless transmission and modeling in mobile cloud computing," *Mobile Networks and Applications*, vol. 18, no. 1, pp. 148–155, 2013.
[4] L. Wang, Y. Cui, I. Stojmenovic, X. Ma, and J. Song, "Energy efficiency on location based applications in mobile cloud computing: a survey," *Springer Computing*, 2013, online DOI: 10.1007/s00607-013-0334-0.
[5] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3g networks," in *IMC 2010*.
[6] J. Yeh, J. Chen, and C. Lee, "Comparative analysis of energy-saving techniques in 3gpp and 3gpp2 systems," *IEEE Transactions on Vehicular Technology*, vol. 58, no. 1, pp. 432–448, 2009.
[7] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Top: Tail optimization protocol for cellular radio resource allocation," in *IEEE International Conference on Network Protocols (ICNP) 2010*.
[8] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *IMC 2009*.
[9] H. Lagar-Cavilla, K. Joshi, A. Varshavsky, J. Bickford, and D. Parra, "Traffic backfilling: subsidizing lunch for delay-tolerant applications in umts networks," *ACM SIGOPS Operating Systems Review*, vol. 5, no. 3, pp. 77–81, 2012.
[10] H. Liu, Y. Zhang, and Y. Zhou, "Tailtheft: Leveraging the wasted time for saving energy in cellular communications," in *Proceedings of the sixth international workshop on MobiArch*. ACM, 2011, pp. 31–36.
[11] A. Pathak, Y. Hu, and M. Zhang, "Fine grained energy accounting on smartphones with eprof," in *EuroSys 2012*.
[12] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Periodic transfers in mobile applications: network-wide origin, impact, and optimization," in *WWW 2012*.
[13] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling," in *MobiCom 2010*.
[14] M. Ra, J. Paek, A. Sharma, R. Govindan, M. Krieger, and M. Neely, "Energy-delay tradeoffs in smartphone applications," in *MobiSys 2010*.
[15] 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 *Mini Infocom 2013*.
[16] A. Rahmati and L. Zhong, "Context-for-wireless: context-sensitive energy-efficient wireless data transfer," in *MobiSys 2007*.
[17] W. Zhang, Y. Wen, and D. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Mini Infocom 2013*.
[18] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *SIGMETRICS 2013*.
[19] M. R. Garey and D. S. Johnson, "Computers and intractability: A guide to the theory of np-completeness," *WH Freeman & Co., San Francisco*, pp. 61–62, 1979.
[20] Y. Cui, S. Xiao, X. Wang, M. Li, H. Wang, and Z. Lai, "Performance-aware energy optimization on mobile devices in cellular network," Department of Computer Science, Tsinghua University, Tech. Rep., 2013. [Online]. Available: http://www.4over6.edu.cn/others/PerES.pdf
[21] M. J. Neely, "Stochastic network optimization with application to communication and queueing systems," *Synthesis Lectures on Communication Networks*, vol. 3, no. 1, pp. 1–211, 2010.