eTime: Energy-Efficient Transmission between Cloud and Mobile Devices

Peng Shu[1,2] Fangming Liu*[1,2] Hai Jin[1,2] Min Chen[2] Feng Wen[1,2] Yupeng Qu[1,2] Bo Li[3]
[1]Key Laboratory of Services Computing Technology and System, Ministry of Education,
[2]School of Computer Science and Technology, Huazhong University of Science and Technology, China.
[3]Hong Kong University of Science and Technology, Hong Kong.

***Abstract*—Mobile cloud computing, promising to extend the capabilities of resource-constrained mobile devices, is emerging as a new computing paradigm which has fostered a wide range of exciting applications. In this new paradigm, efficient data transmission between the cloud and mobile devices becomes essential. This, however, is highly unreliable and unpredictable due to several uncontrollable factors, particularly the instability and intermittency of wireless connections, fluctuation of communication bandwidth, and user mobility. Consequently, this puts a heavy burden on the energy consumption of mobile devices. Confirmed by our experiments, significantly more energy is consumed during "bad" connectivity. Inspired by the feasibility to schedule data transmissions for prefetching-friendly or delay-tolerant applications, in this paper, we present *eTime*, a novel *E*nergy-efficient data *T*ransm*I*ssion strategy between cloud and *M*obile d*E*vices, based on Lyapunov optimization. It aggressively and adaptively seizes the timing of good connectivity to prefetch frequently used data while deferring delay-tolerant data in bad connectivity. To cope with the randomness and unpredictability of wireless connectivity, *eTime* only relies on the current status information to make a global energy-delay tradeoff decision. Our evaluations from both trace-driven simulation and real-world implementation show that *eTime* can be applied to various popular applications while achieving 20%-35% energy saving.**

I. Introduction

With the explosive growth of mobile devices in recent years, users' preference is turning from traditional cell phones and laptops to smartphones and tablets. The increasing portability and capability of mobile devices, together with wide spread of 3G/4G networks and WiFi accesses, have brought rich mobile application experiences to end users. Undoubtedly, the demand for ubiquitous access to a wealth of media content and services will continue to skyrocket.

However, the resource-constrained nature of mobile devices, especially battery life, has been a stumbling block for further improvement on service qualities. To ease this conflict, the paradigm of mobile cloud computing is introduced to extend the capabilities of mobile devices, by taking advantage of abundant cloud resources to help gather, store, and process data for the mobile devices. For example, MAUI [1] and CloneCloud [2] offload parts of resource intensive tasks to the cloud to prolong the battery life and speedup the application. Dropbox extends the storage capacity of mobile devices by storing and synchronizing the mobile data in Amazon S3 storage system. Despite these potential benefits, one of the underpinning components of the whole mobile cloud framework—the interaction between mobile devices and cloud—is usually influenced by uncontrollable factors, in terms of the instability and intermittency of wireless networks. Not only the availability and quality of access points vary from place to place, but also the unlink and downlink bandwidth fluctuates frequently due to multiple factors (*e.g.*, weather, building shield, mobility, flash crowd, etc.).

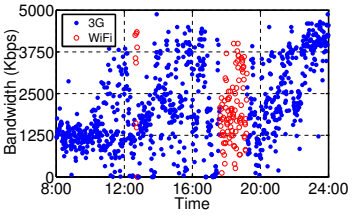

Fig. 1: The measured download bandwidth of a typical Android smartphone using 3G or WiFi in a day.

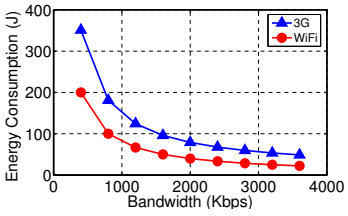

Fig. 2: The energy consumed for downloading 10 MB of data using 3G and WiFi under different achievable bandwidth.

To exemplify the above challenges, we measure wireless bandwidth statistics under representative scenarios in the city of Wuhan, China. Specifically, during an 18-hour period (8:00 - 24:00, July 22, 2012), we traveled around the downtown, carrying an HTC Desire Phone equipped with WiFi and Unicom 3G/HSDPA interfaces. The phone can only use one interface at a time, and it prioritizes WiFi if available. We measured the downlink bandwidth every 60 seconds by downloading 1 MB of data from SAE (Sina App Engine [3]). As shown in Fig. 1, the bandwidth of both 3G and WiFi varies considerably over time and is highly *unpredictable*.

To demonstrate the impact of bandwidth fluctuation on energy consumption, we employ the same smartphone to download 10 MB of data from SAE at different download rates, using 3G and WiFi respectively. As shown in Fig. 2, the energy consumption of transmitting a fixed amount of data is inversely proportional to the available bandwidth condition. It implies that transmitting data in bad connectivity could consume much more energy than that in good connectivity.

*The Corresponding Author is Fangming Liu (fmliu@hust.edu.cn). The research was support in part by a grant from The National Natural Science Foundation of China (NSFC) under grant No.61103176 and No.61133006, by a grant from the Research Fund of Young Scholars for the Doctoral Program of Higher Education, Ministry of Education, China, under grant No.20110142120079, by the CHUTIAN Scholar Project of Hubei Province. Dr. Bo Li's work was supported by a grant from NSFC/RGC under the contract N_HKUST610/11, by grants from HKUST under the contract RPC11EG29, SRFI11EG17-C and SBI09/10.EG01-C, by a grant from Guangdong Bureau of Science and Technology under the contract GDST11EG06, by a grant from ChinaCache Int. Corp. under the contract CCNT12EG01.

For instance, if one happens to watch the news when his/her smartphone suffers from poor connectivity, he/she will not only experience poor service qualities, but also drain the battery life as superfluous energy is consumed. A report from Cisco [4] predicts that, traffic from mobile devices will exceed traffic from wired devices by 2014 and account for 60% of the total IP traffic by 2016. With such a massive amount of data exchanged between mobile devices and the cloud, how to design an energy-efficient transmission strategy for the resource-constrained mobile devices would be a big challenge.

Fortunately, many applications are naturally prefetching-friendly or delay-tolerant, so that data transmission can be scheduled flexibly, *e.g.,* many users may rely on Google Map when visiting new places. Before they actively open the Google Map on their phones to request local maps, the cloud could pull potentially useful maps from the content/application provider via wired and high-bandwidth network backbone and then pre-push them to the phones during periods of good connectivity for energy saving. This is mostly noticed when visiting nature sights, as network connectivity in the wild is not reliable, and battery recharge is infeasible. Similarly, for delay-tolerant applications such as Social Network Service (SNS), one may not immediately check his/her Facebook or Twitter when his/her friends update certain contents, and there is often a time interval between the current and next browsing time. Hence, there exist opportunities for energy saving by deferring the transmission of fresh SNS contents until the 3G/WiFi connectivity becomes satisfactory.

To realize the potential of energy savings in prefetching-friendly or delay-tolerant applications, we present *eTime*, an *E*nergy-efficient data *T*ransm*I*ssion strategy between cloud and *M*obile d*E*vices, which schedules the timing of both uplink and downlink data transmissions. Specifically, *eTime* employs the storage and bandwidth resources in the cloud to help manage data for mobile applications. *eTime* aggressively and adaptively seizes good wireless connectivity of 3G or WiFi to prefetch frequently used data while deferring delay-tolerant data for energy saving. To cope with the intrinsically stochastic characteristics of wireless networks, *eTime* applies Lyapunov optimization [5] techniques to make *online transmission decisions*, which only requires the information of current network bandwidth and data queue backlogs to balance the tradeoff between energy consumption of mobile devices and delays of heterogenous applications. We implement *eTime* on a commercial cloud platform (Sina App Engine) and HTC smartphones running Android Gingerbread OS. Experimental results show that *eTime* can achieve 20%-35% energy savings in various SNS applications.

II. Problem Formulation

We consider a mobile user who has N heterogeneous applications $\mathcal{N} = \{0, 1, 2, \ldots, N\}$ running in the cloud. The data (content) generated in each application is processed in a corresponding queue, denoted as $\mathbf{Q}(t) \triangleq (Q_1(t), \ldots, Q_N(t))$, which is assumed to operate in a discrete time-slotted manner, *i.e.*, $t = 0, 1, 2, \ldots$ For each application i, $Q_i(t)$ represents its queue backlog of data to be transmitted from the cloud to the mobile device at the beginning of time slot t. In every time slot t, we denote the amount of newly generated data of each application as random variables $\mathbf{A}(t) = (A_1(t), \ldots, A_N(t))$, which is alternatively referred to as the vector of data arrival rates that arrive in their corresponding queues $\mathbf{Q}(t)$. We assume that each random variable $A_i(t)$ is i.i.d. over time slots with the expectation of $\mathbb{E}\{\mathbf{A}(t)\} = \boldsymbol{\lambda} \triangleq (\lambda_1, \ldots, \lambda_N)$. Since the workload in a mobile application is dynamic and unpredictable (*e.g.*, a user may open the Google Maps to see a map at a random time), *our model does not assume any a priori knowledge of the statistics* of $A_i(t)$, $\forall i$, $\forall t$.

Let $\omega(t)$ denote the connectivity condition between the cloud and the mobile device in time slot t, and $\omega(t)$ is stochastic and unpredictable as the connectivity varies due to the following reasons: (1) a typical smartphone usually has multiple available links such as a persistent 3G interface and several WiFi access points (AP) but can only activate one of them at a time, thus the actual connectivity depends on the quality of the selected link, (2) even after a particular link is selected, the connectivity remains unstable as it is affected by the user mobility, the flash crowd in the selected link, the limited coverage of the WiFi AP, and even a bad weather, etc.

Therefore, $\omega(t)$ is an uncontrollable random event in our system. We specify $\omega(t)$ as the achievable downlink bandwidth of the selected link in time slot t, as bandwidth is the most critical factor in the energy consumption of data transmission [6]. As a typical mobile device can only use one interface at a time, $\omega(t) \in \{B_{3G}(t), B_{WiFi}(t)\}$, where $B_{3G}(t)$ represents the 3G bandwidth, $\omega(t) = B_{3G}(t)$ if the phone is linked by the 3G interface in time slot t, and $B_{WiFi}(t)$ for the WiFi interface. Choosing 3G or WiFi and selecting the best WiFi AP from all available ones are discussed in previous studies [7]–[9]. Note that these link selection algorithms can be complementary to ours, since given a better link, we will have a higher probability to obtain the good connectivity. In this paper, we adopt a simple and widely used link selection algorithm proposed in [9], in which the mobile device selects the link with the best connection quality by running a series of probe-based tests to the cloud. Based on the selected link, *eTime* further focuses on deciding the proper timing (time slots of higher bandwidth) to transfer data.

Based on the current downlink bandwidth $\omega(t)$ of a mobile device and the queue backlog $\mathbf{Q}(t)$ maintained by the cloud, *eTime* makes a simple decision on $\alpha(t) \in \Omega = \{\text{“}Transmit\ \ Q_i(t)\text{”}, \text{“}Idle\text{”}\}$, where $\alpha(t) = \text{“}transmit\ Q_i(t)\text{”}$ if we decide to transmit the data queuing in $Q_i(t)$ of application i in time slot t, or $\alpha(t) = \text{“}Idle\text{”}$ if we defer the data transmission for energy saving purpose.

For each application i, let $b_i(t)$ denote the amount of data transmitted between the cloud and the mobile device in time slot t, which is determined by the current downlink bandwidth $\omega(t)$ and transmission decision $\alpha(t)$:

$$b_i(t) = \begin{cases} \omega(t)\tau, & \text{if } \alpha(t) = \text{“}Transmit\ Q_i(t)\text{”}, \\ 0, & \text{if } \alpha(t) = \text{“}Idle\text{”}, \end{cases} \quad (1)$$

where τ represents the time span of one time slot.

Denote the energy consumption caused by data transmission on a mobile device as $P(t)$ in time slot t, which also depends the current downlink bandwidth $\omega(t)$ and the transmission decision $\alpha(t)$, *i.e.*, defined by a function $P(\omega(t), \alpha(t))$. Over a long time period T, let M denote the total amount of data transmitted by *eTime*, then $M = \sum_{t=0}^{T-1} \sum_{i=1}^{N} b_i(t)$; correspondingly, the total energy consumption of the mobile device for downloading such an amount of data can be denoted as $E_{eTime} = \sum_{t=0}^{T-1} P(t)$. Specifically, according to practically measured energy consumption models of 3G and WiFi on modern smartphones [6], [10], we can denote $P(t)$ as follows:

$$P(t) = \begin{cases} P_{3G}\tau + P_{tail}t_{interval}, & \text{if } \alpha(t) \neq \text{“}Idle\text{”} \\ & \text{and } \omega(t) = B_{3G}(t), \\ P_{WiFi}\tau, & \text{if } \alpha(t) \neq \text{“}Idle\text{”} \\ & \text{and } \omega(t) = B_{WiFi}(t), \\ 0, & \text{otherwise.} \end{cases} \tag{2}$$

It implies that the energy consumption is decided by the type of interface selected. For WiFi interface, since we assume that a preferable AP has already been selected based on our above mentioned link selection algorithm, the scanning and initial cost is excluded in our model, and the transmission energy via WiFi is proportional to the transmission time. For 3G interface, after transmitting a packet, instead of transiting from the high to low power state immediately, the 3G interface spends substantial time in the high power state to avoid the delay and signaling overhead incurred due to frequent power state transition, channel allocation and release. This mechanism incurs considerable *tail energy* [10], [11]. Since lingering in the high power state consumes more energy, a fixed tail time T_{tail} is empirically used to set the inactivity timer according to the performance-energy tradeoff [10]. Let $t_{interval}$ denote the time interval between the current transmission and the last transmission, then we have $t_{interval} \in (0, T_{tail})$.

Based on the above system model, we define the time-averaged energy consumption of the mobile device as

$$\overline{P} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \mathbb{E}\{|P(t)|\}. \tag{3}$$

According to our experimental observations in Fig.1 and Fig.2, an intuitive strategy for energy minimization is to transmit data only when the 3G or WiFi connectivity (downlink bandwidth) is good enough. However, if we greedily defer data transmissions for aggressive energy saving consideration, the queue backlog $\mathbf{Q}(t)$ of all applications will increase unboundedly, consequently leading to large unacceptable delays and poor user experiences. Hence, there is an *energy-delay tradeoff* in scheduling the data transmissions.

To balance such a tradeoff, we require all the queues to be stable in the time average sense, i.e.,

$$\overline{\mathbf{Q}} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=0}^{T-1} \sum_{i=1}^{N} \mathbb{E}\{|Q_i(t)|\} < \infty, \tag{4}$$

where $\overline{Q}$ represents the *time-averaged queue backlog* and the queueing dynamics can be characterized by:

$$Q_i(t+1) = \max(Q_i(t) - b_i(t), 0) + A_i(t). \tag{5}$$

Condition (4) implies that all the data achieved at the queues in the cloud will be transmitted to the mobile devices in bounded time. A larger value of $\overline{Q}$ implies a longer delay for all the applications.

In each time slot t, *eTime* makes an online transmission decision subject to $\alpha(t) \in \Omega$, with the objective of minimizing the time averaged energy consumption under queue stability constraints for all applications:

$$\text{minimize} \quad \overline{P} \triangleq \limsup_{T\to\infty} \frac{1}{T} \sum_{t=1}^{T-1} \mathbb{E}\{|P(t)|\} \tag{6}$$

$$\text{subject to} \quad \overline{\mathbf{Q}} < \infty,\ \alpha(t) \in \Omega. \tag{7}$$

III. Design and Analysis of Energy-Efficient Transmission Strategy

Since wireless networks are highly stochastic and unpredictable, we can only take advantage of the information of the current downlink bandwidth to make a smart transmission decision. Our considerations on energy efficiency and queue stability in the stochastic networks lead us to a design approach based on the Lyapunov optimization framework [5], which has been widely used in energy consumption optimization problem [7], [12]. Most importantly, the framework enables us to only utilize the knowledge of current states (*i.e.,* network bandwidth and data queue backlogs) to approach problem (6).

A. Problem Analysis Using Lyapunov Optimization

We first define a Lyapunov function, $L(\mathbf{Q}(t))$, which represents a scalar metric of queue congestion [5] for reflecting delays of applications, as follows:

$$L(\mathbf{Q}(t)) \triangleq \frac{1}{2} \sum_{i=1}^{N} {Q_i}^2(t), \tag{8}$$

where $L(\mathbf{Q}(t)) \geq 0$, $\forall t$. A smaller value of $L(\mathbf{Q}(t))$ implies that all the queue backlogs are small, while a larger value of $L(\mathbf{Q}(t))$ implies that at least one queue (for one application) is congested. To keep the system stable by persistently pushing the Lyapunov function towards a lower congestion state, we next introduce the Lyapunov drift $\Delta(\mathbf{Q}(t))$ as follows:

$$\Delta(\mathbf{Q}(t)) \triangleq \mathbb{E}\{L(\mathbf{Q}(t+1)) - L(\mathbf{Q}(t))|\mathbf{Q}(t)\}, \tag{9}$$

which is the expected change in the Lyapunov function over one time slot, given that the current state in time slot t is $\mathbf{Q}(t)$. Following the Lyapunov optimization approach [5], we incorporate the expected energy consumption over one time slot, to both sides of Eq. (9), which leads to *drift-plus-penalty* term: $\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{P(t)|\mathbf{Q}(t)\}$.

The control parameter $V > 0$ represents a *design knob of the energy-delay tradeoff*, *i.e.,* how much we shall emphasize the energy minimization (6) compared to transmission delays (7). It empowers *eTime* to make flexible design choices between

transmission delay and energy consumption according to the application type and user context, *e.g.,* when the mobile device is running out of power, the user may allow longer delay of the data synchronization in Dropbox for energy conservation. *eTime* can handle this situation by setting a larger value of V.

Then, a key derivation step is to obtain an upper bound on this term. The following lemma defines such an upper bound for our case.

Lemma 1: Given any possible wireless bandwidth $\omega(t)$, data queue backlogs $\mathbf{Q}(t)$ and data arrival rates $\mathbf{A}(t)$, under any possible decision $\alpha(t)$, we have:

$$\Delta(\mathbf{Q}(t)) + V\mathbb{E}\{P(t)|\mathbf{Q}(t)\} \leq B + V\mathbb{E}\{P(t)|\mathbf{Q}(t)\} + \sum_{i=1}^{N}\mathbb{E}\{Q_i(t)(A_i(t) - b_i(t))|\mathbf{Q}(t)\}. \quad (10)$$

where $B = (A_{max}^2 + b_{max}^2)/2$, and $A_{max} \geq A_i$, $\forall i \in \mathcal{N}$, represents the maximum amount of data that can arrive for any application i per time slot and $b_{max} \geq b_i$, $\forall i \in \mathcal{N}$, represents the maximum amount of data that can be transmitted via the wireless network in a time slot. Proof (pp. 33, [5]).

Following the design principle of Lyapunov framework, the underlying objective of our optimal transmission decision is to minimize the upper bound of the drift-plus-penalty term, *i.e.*, in every time slot t, we try to choose a control action $\alpha(t)$ to minimize the RHS of Eq. (10). Note that the transmission decision $\alpha(t)$ can only affect the energy consumption $P(t)$ and the data transmitted in a time slot $b_i(t)$. By taking use of the concept of opportunistically minimizing an expectation, our algorithm finally minimizes the RHS of Eq. (10) by minimizing the following simplified term:

$$VP(t) - \sum_{i=1}^{N} Q_i(t)b_i(t). \quad (11)$$

Specifically, our algorithm is described in Algorithm 1, which makes a decision $\alpha(t)$ drawn from the finite set $\Omega \triangleq \{$"*Transmit* $Q_i(t)$", "*Idle*"$\}$ in each time slot t.

Algorithm 1: *Energy-efficient transmission strategy*

1: At the beginning of each time slot t, monitor the queue backlog $Q_i(t)$ of each application i and the real-time bandwidth $\omega(t)$ between cloud and the mobile device.
2: Determine the transmission decision $\alpha(t)$ that minimize (11), where $P(t)$ and $b_i(t)$ are given by (2) and (1).
3: If $\alpha(t) =$ "*Transmit* $Q_i(t)$", download the data in application i from the cloud to the mobile device; otherwise, stay idle for energy conservation.
4: Update queues $\mathbf{Q}(t+1)$ in the cloud using (5) when the current time slot t ends.

Remarks: Algorithm 1 implies several straightforward intuitions. Considering a fixed V, if the control action $\alpha(t) =$ "Idle", we choose not to transmit any data in slot t, so we have $P(t) = 0$ and $b_i(t) = 0$, $i = 1, 2, \ldots, N$, then the expression $VP(t) - \sum_{i=1}^{N} Q_i(t)b_i(t) = 0$. As we only execute the control action which minimizes the expression $VP(t) - \sum_{i=1}^{N} Q_i(t)b_i(t)$, the transmission takes place only if some control action $\alpha(t)$ could satisfy $VP(t) - \sum_{i=1}^{N} Q_i(t)b_i(t) < 0$. It happens when either the bandwidth $\omega(t)$ is high, making a large $b_i(t)$, or some queue $Q_i(t)$ is already congested in time slot t. Therefore, our algorithm will choose to transmit data in the following conditions: (1) when the connectivity is good enough, the mobile devices will catch the chance to transmit more data with less energy, (2) when certain queues in the cloud are congested, the data transmission must be triggered to guarantee the queue stability.

B. Performance Bounds

The performance bounds of *eTime* transmission algorithm are stated in the following theorem.

Theorem 1: Assume that the data arrival rate $\boldsymbol{\lambda}$ is strictly within the network capacity region Λ, then under Algorithm 1, the performance bounds of the time average energy consumption and queue backlog can be denoted as:

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T-1}\mathbb{E}\{|P(t)|\} < P^* + \frac{B}{V}, \quad (12)$$

$$\limsup_{T\to\infty} \frac{1}{T}\sum_{t=1}^{T-1}\sum_{i=1}^{N}\mathbb{E}\{|Q_i(t)|\} < \frac{B + VP^*}{\epsilon}, \quad (13)$$

where $\epsilon > 0$ represents a measure of the distance between the data arrival rate $\boldsymbol{\lambda}$ and the network capacity region Λ, and P^* is the lower bound of $\overline{P}$. Proof (pp. 47, [5]).

Remarks: Eq. (12) and Eq. (13) demonstrate an $[O(1/V), O(V)]$ energy-delay tradeoff. We can use an arbitrarily large V to make the energy cost $\overline{P}$ arbitrarily close to the optimum P^*. However, this comes with a cost, as Eq. (13) implies that the average queue backlog $\overline{Q}$ grows linearly with V. Such an energy-delay tradeoff allows *eTime* to make flexible design choices according to different application types and user contexts, and find the sweet spot of V such that further increasing V yields very small reduction in $\overline{P}$.

IV. Performance Evaluation

A. Trace-driven Simulation

Parameter Setting: To evaluate how *eTime* performs under realistic mobile networking conditions, we use a network trace gathered in Wuhan, China. Trace-driven simulation provides experimental repeatability for us to study the impact of parament V for a best selection.

According to Algorithm 1, we need to estimate the achievable bandwidth $\omega(t)$ at the beginning of every time slot t, this estimation will lead to very high overheads if the length of slot τ is too small. However, if τ is too large, it's unconvincing to let $\omega(t)$ represent the network condition over the whole slot t, as the bandwidth may vary widely on an extended slot. Therefore, according to observed bandwidth statistics in our trace, we set τ as a moderate value (60 seconds). Every 60 seconds, we use the downlink bandwidth in our trace as the bandwidth estimation $\omega(t)$ for the corresponding time slot t.

To simulate applications with different volumes of data, we determine data arrival rates according to the data patterns of traditional applications on mobile phones. We consider 10 applications running in the cloud and assume that their data arrivals all follow *Poisson Processes* with $\mathbb{E}\{A_i(t)\} = \lambda_i = \lambda$ for all i, where λ is chosen from $\{0.25, 0.5, 1.0, 2.0\}$ packets/minute. The size of one packet is 100 KB. We run our algorithm in $1,000$ time slots for each of the V value ranging from 1 to 500. According to the power models developed in [6], [10] and our measurements on HTC Desire, we set the power coefficients P_{3G}, P_{tail}, and P_{WiFi} to 1.5 W, 0.7 W and 1.0 W, respectively.

Simulation Results: The performance of *eTime* depends on parameter V, which controls the energy-delay tradeoff. As shown in Fig. 3, given the arrival rate $\lambda = 1.0$, as V goes from 1 to 500, the time-averaged energy consumption drops from 76 J to 7 J, and the time-averaged queue backlog grows from near 0 to $1,100$. Note that energy consumption falls quickly at the beginning and then tends to descend slowly while the time-averaged queue backlog grows linearly with V. This finding confirms the $[O(1/V), O(V)]$ energy-delay tradeoff as captured in Eq. (12) and (13). Particularly, there exists a *sweet spot* of V (*e.g.,* $V = 100$ when $\lambda = 1.0$), beyond which increasing V leads to marginal energy conservation yet consistently growing delays.

B. Real-World Implementation

Experimental Setup: We develop a prototype of *eTime* in about 1, 000 lines of PHP code and deploy it on Sina App Engine (SAE), a cloud platform that provides Platform as a Service (PaaS). On the mobile device side, we develop a mixed social networking application, *AllFriends*, in about 5, 000 lines of Java code. *AllFriends* employs *eTime* to manage all the contents shared from one's various SNS accounts (*e.g.,* Facebook, Twitter, Sina Weibo, RenRen), and pre-pushes the content to his/her mobile phone.

We install *AllFriends* into our HTC Desire smartphone and frequently use it as a common SNS application over several days. We record the amount of data ($M = \sum_{t=0}^{T-1} \sum_{i=1}^{N} b_i(t)$) transmitted via *eTime* and the energy consumption $E_{eTime} = \sum_{t=0}^{T-1} P(t)$ per day. We compare E_{eTime} with the random strategy E_{random}: the user may open applications (which run only on mobile devices) and request data in a random time slot, and the data are then transmitted instantly from content providers to the mobile device under unpredictable wireless connectivity. To estimate E_{random}, we use the average network condition observed in the past as a reasonable approximation, i.e., we derive E_{random} as $E_{random} = \frac{M}{\overline{B_{WiFi}}} \times P_{WiFi} \times A_{WiFi} + (\frac{M}{\overline{B_{3G}}} \times P_{3G} + T_{tail} \times P_{tail}) \times (1 - A_{WiFi})$, where $\overline{B_{3G}}$ and $\overline{B_{WiFi}}$ represent the average bandwidth of 3G and WiFi, respectively, and A_{WiFi} represents the average observed availability of WiFi.

Experimental Results: The data arrival rate in SNS application is mainly determined by the number of a user's fellows. We use the accounts with the total number of one's fellows ranging from 200 to 800. We record the energy consumption and data size in days, and compare the energy consumption with the random strategy. From Fig. 4, we can see that *AllFriends* consumes less energy (19%-37%) in transmitting the same amount of contents. What's more, as the contents have already been stored before opening the application, the user can experience zero latency when checking his/her SNS.

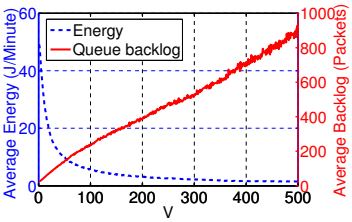

Fig. 3: The impact of V on time-averaged energy consumption and queue backlog.

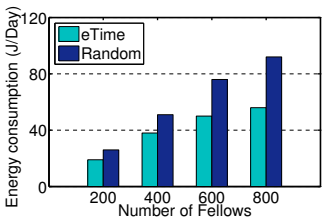

Fig. 4: Energy consumption vs. fellow magnitudes under different strategies.

V. Conclusion and Future Work

In this paper, we present *eTime*, which achieves energy-efficient data transmission in mobile cloud computing by taking advantage of cloud resources to manage data for prefetching-friendly or delay-tolerant applications. Based on Lyapunov optimization, *eTime* adaptively makes transmission decisions according to real-time network conditions and data queue backlogs. Through both trace-driven simulations and real-world implementation, we show that notable energy conservation can be achieved on mobile devices with *eTime*. For future work, a mobile-cloud middleware will be developed to apply *eTime* to unmodified mobile applications seamlessly.

References

[1] E. Cuervo, A. Balasubramanian, D. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "Maui: making smartphones last longer with code offload," in *Prof. of ACM MobiSys*, Jun. 2010.

[2] B. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "Clonecloud: Elastic execution between mobile device and cloud," in *Prof. of ACM EuroSys*, Apr. 2011.

[3] Sina App Engine. [Online]. Available: http://sae.sina.com.cn/

[4] Cisco Visual Networking Index: Forecast and Methodology, 2011-2016. [Online]. Available: http://www.cisco.com/

[5] M. Neely, "Stochastic network optimization with application to communication and queueing systems," *Morgan & Claypool Publishers*, 2010.

[6] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. Dick, Z. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Prof. of eighth IEEE/ACM/IFIP international conference on Hardware/software codesign and system synthesis*, 2010.

[7] M. Ra, J. Paek, A. Sharma, R. Govindan, M. Krieger, and M. Neely, "Energy-delay tradeoffs in smartphone applications," in *Prof. of ACM MobiSys*, 2010.

[8] D. Gupta, P. Mohapatra, and C. Chuah, "Seeker: A bandwidth-based association control framework for wireless mesh networks," *Wireless Networks*, pp. 1–18, 2011.

[9] A. Nicholson, Y. Chawathe, M. Chen, B. Noble, and D. Wetherall, "Improved access point selection," in *Prof. of ACM MobiSys*, 2006, pp. 233–245.

[10] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network applications," in *Prof. of IMC*, Nov. 2009.

[11] A. Pathak, Y. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Prof. of ACM EuroSys*, 2012.

[12] 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 *Prof. of IEEE Infocom*, Mar. 2012.