Optimizing Offline Access to Social Network Content on Mobile Devices

Ngoc Do[1], Ye Zhao[1], Shu-Ting Wang[2], Cheng-Hsin Hsu[2], and Nalini Venkatasubramanian[1]
[1]Department of Information and Computer Science, University of California, Irvine, USA
[2]Department of Computer Science, National Tsing Hua University, Hsin-Chu, Taiwan

***Abstract*—In this paper, we explore the problem of supporting efficient access to social media contents on social network sites for mobile devices without requiring mobile users to be online all the time. We propose and implement a broker/proxy based architecture that stages data at a broker/proxy, and selectively downloads to the mobile device only those contents that have a high likelihood of being viewed. The system determines the relevance of social media updates that continuously arrive (e.g., Facebook friend updates) for each user. Using knowledge of this relevance and current network/system conditions, we develop scheduling algorithms that determine which social contents are sent to the devices. We develop an Android app providing offline access to Facebook. Our experimental results indicate that our system is energy efficient, which saves energy by 6.9 times for WiFi and 9.1 times for cellular connections. We also use data traces gathered from our app to further drive extensive simulation based evaluations which show that our proposed algorithms provide efficient facilities for tuning the system's performance.[1]**

***Index Terms*—Social networks, mobile devices, optimization, energy consumption, offline access.**

I. Introduction

The phenomenal popularity of social networks, such as Facebook, Twitter, LinkedIn, Google+, and Instagram, has changed the way people interact today. Indeed, many people rely on these social networks to communicate with their friends, family, and community on a day to day basis. The ability to continue these interactions anytime, anywhere seamlessly is quickly becoming commonplace, and users on modern mobile devices expect to not just to access social networks but also exchange rich media contents, such as video, audio, and images, for an enhanced user experience. It is reported that 93% of Android smartphone users in India use social networks on their smartphones [1] and often this is the reason why they purchase smartphones in the first place. In North America, a recent IDC report on smartphone users indicates that 70% of them access Facebook via smartphones, and more strikingly, 40% of users feel *connected* when using Facebook, only trailing 43% for making voice calls and 49% for texting [2]. In fact, the main finding of the IDC report is "mobile+social=connectedness", i.e., people feel isolated without mobile access to social networks.

To ensure this constant connectedness, mobile users subscribe (and pay for) 3G/4G data plans that are often expensive and do not work for a host of reasons including: (a) wireless network availability is sporadic (accessibility of WiFi access points, unpredictable data rates in 3G networks), (b) mobile devices are battery-powered with stringent energy budgets that are easily depleted from constant connectivity to and interaction with WiFi/3G networks, (c) the shared network bandwidth is limited in public locations (where users want to access this information), and (d) dataplans are becoming volume-driven and hence costly.

We observe that the basic need (root cause) of always-on connectedness stems from the assumption that existing mobile social mobile apps, such as mobile Facebook app, *expect* always-on connectivity. The modus operandi today is that these apps synchronize with social networks when mobile users launch the apps and mobile devices are connected to the Internet. We believe that offering offline access for these apps is necessary for mobile users in order to interact with social contents when Internet access is not available. There are many scenarios in which the offline access feature is useful. For example, a student takes a subway to his or her school. In the subway, the student does not typically have mobile Internet connections to social network sites, and thus the offline access feature is the only option allowing the student to view and interact with updates from his or her friends that are prefetched before the student steps into the subway.

The problem of optimally prefetching social media content from social media providers to mobile users is more complex than it seems at first glance. Multiple factors contribute to the non-uniform nature of the problem at mobile users, contents, and system levels. More specifically, mobile users have different/personalized viewing needs and preferences; social media contents are diverse in size and importance; and network conditions may be highly dynamic due to fading, shadowing, interference, and congestion on wireless/wired links. All of the above factors make prefetching social media contents on mobile devices quite challenging.

In this paper, we envision a distributed broker/proxy solution, in which broker/proxy nodes in the network are employed to help mobile devices to optimally prefetch social media contents. Fig. 1 illustrates the considered broker/proxy architecture where mobile users and social media providers exchange data by staging relevant content intelligently with the supports from the broker/proxy nodes. The goal of the broker/proxy architecture is to maximize the viewing likelihood and quality of experience of the prefetched social media contents while maintaining certain energy level at mobile devices for user daily activities. The main contributions of this paper are:

1) We design a broker/proxy based architecture and framework to prefetch media contents from online social networks and deliver the contents efficiently to mobile users

[1]This research has been supported in part by NSF grants 1063596, 1059436, 1352727 and 1143705.

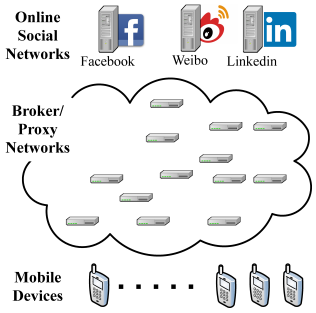

Figure 1. Considered system architecture.

through the broker/proxy nodes (Section II).

2) We develop fine-grained control mechanisms for optimizing data transfer to mobile devices based on the Lyapunov optimization framework that maximizes the probability of content viewing while maintaining a guaranteed residual energy level at mobile devices. We design and develop an optimal scheduling technique and a near-optimal technique (that achieves an approximation factor of 2) using the above framework to deliver media contents to mobile users - our scheme accounts for the relevance of contents to the user and is aware of energy consumption at mobile devices (Section III).
3) We evaluate our system using a testbed of Android phones. We compare our broker/proxy based system against an offline pull based system that executes completely at mobile devices. Our results illustrate the broker/proxy based system saves energy 6.9 times for WiFi connections and 9.1 times for cellular connections. We also collect data traces from our users for simulation based evaluations. The results from the simulations confirm the correctness and efficiency of our proposed algorithms (Section IV).

II. A Broker/Proxy Framework for Social Content Delivery

A straightforward approach to enabling offline social network access is to program mobile devices to periodically check the social networks for new updates and download relevant updates. This approach, however, is expensive both in terms of the consumed data plan quota and mobile device energy, which is caused primarily by the limitations in the current APIs of social network providers. In order to determine whether there are new updates[2], mobile device has to set up connections, and send requests for new updates. If there are no new updates, the requests are useless, and device resources (i.e., energy) consumed for sending the requests are wasteful. In some cases, mobile devices even have to pull/retrieve multiple data items (e.g., posts, comments/likes of Facebook posts), and analyze the data to detect new updates. Executing these processes on mobile devices periodically is taxing on resource-constrained mobile devices. A better approach is to employ a system of brokers

[2]Social network sites, such as Facebook, provide limited notification service, that is only for updates related directly to users, but not for updates from friends, e.g., items in Facebook News Feed page.

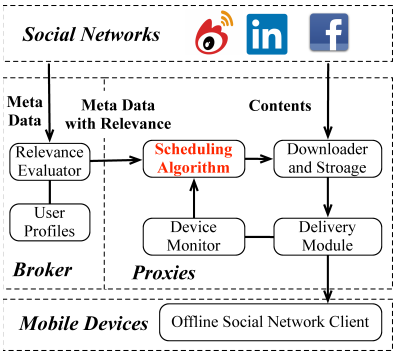

Figure 2. Major software modules in our system.

and proxies in the network, in order to download, analyze, and deliver new updates from social network sites to mobile users.

Our proposed approach consists of a system of brokers/proxies that analyzes new updates and downloads the most relevant content from social network sites to mobile users. In this system, the key coordination is performed at a broker – social media users on mobile devices register with the system through this brokerage service. The broker service periodically checks and tracks social media updates for each mobile user, calculates the relevance of every update to the user and generates an annotated list of updates, enhanced with the calculated relevance factors. A broker may manage one or more proxies, and each proxy in turn serves one or multiple mobile devices. The broker passes the annotated list of new contents to each mobile device's proxy and also initiates the actual downloads of the updated contents onto the respective network proxies (usually, in close vicinity of the client). In general, proxies may be installed by mobile users, cellular carriers, content distribution networks, social networks, and third-party companies.

In this paper, we assume that proxy selection is done by individual mobile users, who may have lists of trustworthy proxies in mind or prefer to use closer proxies for higher network bandwidth. More sophisticated policies for proxy selection are possible [3] - for example multiple brokers/proxies may be employed for scalability and mobility handling - the design of such policies are out of the scope of this paper. We aim to design, implement, and evaluate a broker/proxy solution that schedules mobile prefetches to achieve these goals and relieve the mobile device from the associated computation and communication overhead.

Fig. 2 illustrates the software modules of the proposed system. The broker/proxy periodically checks for updates from social network sites and efficiently delivers the updates to mobile users. The crux of the system resides in two key modules that are described below: (i) *user-content relevance evaluator*, which estimates the likelihood of a content being watched by a user (based on social connections) and (ii) *scheduling algorithm*, which performs the actual delivery of contents to mobile users such that the benefits of prefetching contents is maximized under the resource constraints.

User-content relevance evaluator. We present only briefly here how we evaluate user-content relevance using a learning technique. The reader can find more details in our previous

work [4]. This module estimates the likelihood a social content will be viewed by a user using social network information. The resulting user-content relevance provides hints to the scheduling and delivery modules in the proxy for resource-efficient delivery. Existing approaches to calculate content relevance in social networks use: (1) content based approaches [5]–[7] that exploit correlation among content attributes such as keywords, (2) collaborative filtering approaches [8]–[10] that capture user similarity and use this to determine the user interest in a content based on similar users' interests on the same content, and (3) social network based approaches [11]–[13] that exploit social relations to determine whether a user is likely to view a content. We follow the approaches [11]–[13] in the last category to rank social media contents based on social relations and interactions. However, unlike prior techniques, our relevance evaluation scheme is designed to have low complexity and be scalable to frequent arrival of new contents.

There are two steps in our relevance evaluation scheme: (a) *social-based feature construction* and (b) *viewing likelihood prediction*. In the first step, we construct social features and interactions to capture user interests on a content. Four features of mobile social networks are constructed:

1) *Post interactions*, which capture the interactions between a user and his/her friends via posts, comments, and likes.
2) *Private message exchanges*, which capture the private message interactions among users.
3) *User interests*, which capture the number of view clicks of a user on contents posted by another user.
4) *Post popularity*, which capture the number of comments and likes of a post for the global popularity of a post.

The user-content relevance evaluator collects all the features and applies a training process in the next step.

The second step employs a logistic regression classifier to *build a prediction model and evaluate the viewing likelihood for a newly arriving media content and user pair*. In particular, the logistic regression classifier evaluates and calculates proper coefficients for the impact of all four features. The learning model is then used to evaluate the viewing likelihood for a newly arriving media content. Our experiments show that the model parameters are calculated in a lightweight and quick manner. The maximum running time to build the model in our experiments is only a few hundreds of milliseconds. Therefore, the user-content relevance evaluator can frequently update the model to reflect current behaviors of the user.

Scheduling algorithm. The second key component is the *Scheduling Algorithm module* that executes on the proxy. The challenge here is to utilize output (relevance) from the relevance evaluator module to efficiently schedule the delivery of contents from the proxy to the mobile device based on the available resources at the mobile device, e.g., network bandwidth and energy level. Updates from social network sites for a user are downloaded to the selected proxy via the *Download and Storage module*. Decisions are made by the scheduling algorithm running on the proxy to perform the prefetch of relevant updates under good network conditions and battery levels at the mobile device. These system conditions are captured via a *Device Monitor* at the proxy and the *Delivery module* implements the content transfer. The details of the scheduling algorithm are presented in the next section.

III. Content Scheduling Algorithm

In this section, we formulate and solve the scheduling problem of prefetching social contents to mobile device optimally.

A. System Models and Problem Formulation

We divide time into slots. Every time slot, the scheduling algorithm selects which contents will be delivered to mobile device.

Content arrival. A new content c generated by a friend of u is fetched by the proxy and stored in the proxy's database before it can be delivered to the user. Content c includes metadata (e.g., post_id, author_id, description, source links of images and videos, and etc.) and media content (images and videos). Let us denote the size of content c (including metadata and media content) as s_c, and the total size of all contents for the user fetched by the proxy in time slot t as $\lambda(t)$. We have $\lambda(t) = \sum_{c\in a(t)} s_c$, where $a(t)$ is a set of contents arriving to the proxy in t. Assume that $\lambda(t)$ is finite and limited by λ_{max}, i.e., $\lambda(t) \leq \lambda_{max} \forall t$. The time averaged fetched data amount is defined as:

$$\bar{\lambda} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\lambda(\tau)\}. \quad (1)$$

Content departure and dropping. Every time slot, the scheduling algorithm chooses a subset of the fetched contents to deliver to the user. Let $r(t)$ be a set of contents scheduled to be delivered in time slot t, and $\mu(t)$ be the total delivered data amount, i.e., $\mu(t) = \sum_{c\in r(t)} s_c$. We assume $\mu(t) \leq \mu_{max} \forall t$. It is not hard to see that $\mu(t)$ is limited by the mobile device's downlink bandwidth at t, defined as $\theta(t)$, i.e., $\mu(t) \leq \theta(t)$. We also assume $\theta(t) \leq \theta_{max} \forall t$. The time averaged data amount delivered to the user is defined as:

$$\bar{\mu} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\mu(\tau)\}. \quad (2)$$

Some fetched contents are dropped by the proxy due to low viewing probability or large size (hence, high energy consumption). Let $d(t)$ and $\chi(t)$ be a set of dropped contents and the total dropped data amount in time slot t, i.e., $\chi(t) = \sum_{c\in d(t)} s_c$. The time averaged data amount dropped from the fetched content list is:

$$\bar{\chi} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\chi(\tau)\}. \quad (3)$$

Energy budget and consumption. In our system, each user sets an energy threshold e_t and nothing is prefetched if the current energy level, denoted as $e(t)$ is less than e_t. Moreover, we allow mobile users to set energy budget for prefetching contents, in order to preserve battery levels for other daily activities, such as making phone calls and browsing Web pages. The energy budget is set for user-specified time periods, e.g., a user may set the energy budget to 10% of the current energy for a period between 8 a.m. to 6 p.m., and to 30% between 6 p.m. to 8 a.m. In particular, we let t_s and t_e be the starting and ending times of a period. Assume period $[t_s, t_f]$ consists of a set of time slots $\{t_0, t_1, ...\}$. Let us define e_a as the available energy amount at the device at t_s, and e_u as the usable energy percentage for the system to download social contents in the

period. The energy budget at the beginning of the period is $e_p = [(e_a - e_t)e_u]^+$, where $[x]^+ \triangleq \max(0, x)$. For every content download, the consumed energy is deducted from the energy budget. The energy budget may increase if the phone is charged. In time slot t, if the phone is charged and the current energy level is larger than e_t, e_u percentage of the charged energy will be added to the energy budget. Let $w(t)$ be the charged energy in time slot t. The energy added to the budget is $\varepsilon(t) = w(t)e_u$. We write $\varepsilon(t_0) = e_p + w(t_0)e_u$ for t_0, and define the time averaged energy added to the budget as:

$$\bar{\varepsilon} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\varepsilon(\tau)\}. \tag{4}$$

We next model the energy consumed by mobile device for receiving contents from the broker/proxy. Let $\rho_c(t)$ be the energy amount used for receiving content c in time slot t. We employ the energy models from [14] to estimate $\rho_c(t)$, based on the content, network type, and network condition. The total energy consumption in time slot t is $\rho(t) = \sum_{c\in r(t)} \rho_c(t)$, where $r(t)$ is the contents delivered to the mobile device in t. The time averaged energy consumption amount is written as:

$$\bar{\rho} \triangleq \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{\rho(\tau)\}. \tag{5}$$

Utility model. Let p_c be the probability a user u will view c, which is between 0 and 1. The closer to 1 p_c is, the more likely u views c. We consider time decay of contents, i.e., a more recent content interests users more. Let $U_c(t) = f(p_c, t)$ be the time decay utility function that models the benefit to deliver content c to the user in time slot t, which is a decreasing function on p_c and t. In a time slot, some contents are delivered to the user while some are dropped, or the proxy may be totally filled up. Let's denote $U_r(t)$ and $U_d(t)$ as the total utilities of contents which are delivered and dropped, respectively. Mathematically, $U_r(t) = \sum_{c\in r(t)} U_c(t)$ and $U_d(t) = \sum_{c\in d(t)} U_c(t)$.

Problem formulation. The scheduling algorithm selects a set of contents $r(t)$ to deliver to the mobile device and a set of contents $d(t)$ to drop from the proxy based on the following formulation.

$$\max: \quad \bar{U} = \lim_{t\to\infty} \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\Big\{U_r(t) - \alpha U_d(t)\Big\}; \tag{6a}$$

$$\text{st:} \quad \bar{\lambda} \le \bar{\mu} + \bar{\chi}; \tag{6b}$$

$$\bar{\rho} \le \bar{\varepsilon}; \tag{6c}$$

$$\sum_{c\in r(t); r(t)\in f(t)} s_c \le \theta(t); \tag{6d}$$

$$\sum_{c\in r(t); r(t)\in f(t)} \rho_c(t) \le \min(e(t) - e_t, P(t)). \tag{6e}$$

The objective function in Eq. (6a) maximizes the long term utility for content delivery and minimizes the long term utility for content dropping. The proxy drops some contents to stabilize the system as shown in constraint in Eq. (6b). Weight α ($\alpha \ge 1$) is added to the dropping utility to force the system to consider more seriously when making a dropping decision. Constraint Eq. (6c) ensures that the long term energy budget is enough to perform scheduled transfers. Constraint Eq. (6d) ensures that bandwidth is enough for data transfer at any time slot. And the last constraint in Eq. (6e) makes sure that energy is enough for data receipt from the broker/proxy (note that, $e(t) - e_t$ is the difference of the current energy level with the power threshold, and $P(t)$ is the remaining amount in energy budget that will be modeled in the next part).

B. An Optimal Algorithm

We design an algorithm to solve the problem in Eqs. (6a)–(6e) using the fine-grained queue control theory – Lyapunov optimization [15], which is useful for solving long-term maximization problems. We start by presenting our design principles, and then the detailed algorithm.

Algorithm design principles. We use a real queue Q and a virtual queue P to represent constraints in Eqs. (6b) and (6c) for data and energy, respectively. Q stores incoming contents downloaded to the proxy. The contents in Q are taken out of the queue for either delivery or drop. The evolution of Q over time is written as:

$$Q(t+1) \triangleq [Q(t) - \mu(t) - \chi(t) + \lambda(t)]^+. \tag{7}$$

We define the stability of the Q as:

$$\bar{Q} \triangleq \lim_{t\to\infty} \sup \frac{1}{t} \sum_{\tau=0}^{t-1} \mathbb{E}\{Q(\tau)\} < \infty. \tag{8}$$

At $t = 0$, we set $Q(t) = 0$. The constraint in Eq. (6b) is satisfied, as long as Q is stable.

Virtual queue P is just a counter in memory, which indicates the energy budget for the mobile device to prefetch contents. The evolution of P over time is:

$$P(t+1) \triangleq [P(t) - \rho(t) + \varepsilon(t)]^+. \tag{9}$$

At $t = 0$, we set $P(t) = 0$. In Section III-C, we show that $P(t)$ is deterministically upper bounded. This therefore satisfies the constraint in Eq. (6c).

We define vector $\phi(t) = [Q(t); P(t)]$ and a quadratic Lyapunov function:

$$L(\phi(t)) \triangleq \frac{1}{2}\Big[Q(t)^2 + (P(t) - \kappa)^2\Big], \tag{10}$$

where κ is a constant. Intuitively, if the system is stabilized, i.e., $L(\phi(t))$ is maintained small, $P(t)$ is accumulated close to κ. Therefore, by choosing a reasonable κ, we control the available energy amount for the mobile device to prefetch contents. A one-step Lyapunov drift is defined as below:

$$\Delta(\phi(t)) \triangleq \mathbb{E}\{L(\phi(t+1)) - L(\phi(t))|\phi(t)\}. \tag{11}$$

The intuition is if $\Delta(\phi(t))$ is maintained negatively over time, the system is stable.

Lemma 1 (Lyapunov drift): The one-step conditional Lyapunov drift satisfies the following constraint at any time slot under any algorithms used to control the system:

$$\begin{aligned} \triangle(\phi(t)) &= \mathbb{E}\{L(\phi(t+1)) - L(\phi(t))|\phi(t)\} \\ &\le \beta - \mathbb{E}\{Q(t)\mu(t) + (P(t) - \kappa)\rho(t)|\phi(t)\} \\ &\quad - \mathbb{E}\{Q(t)\chi(t)|\phi(t)\} \\ &\quad + \mathbb{E}\{Q(t)\lambda(t)|\phi(t)\} \\ &\quad + \mathbb{E}\{(P(t) - \kappa)\varepsilon(t)|\phi(t)\}, \end{aligned} \tag{12}$$

where β is a carefully chosen constant.

Our goals are to minimize the drift and maximize the system's utility. Thereby, we obtain:

$$\min: \triangle(\phi(t)) - V\mathbb{E}\{U_r(t) - \alpha U_d(t)|\phi(t)\}, \tag{13}$$

where V is a weight that controls how important the utility is, compared to the system stability.

By applying Lemma 1 to Eq. (13), we maximize the sum $\Phi(t) = \Phi_1(t) + \Phi_2(t)$, where:

$$\Phi_1(t) = \mathbb{E}\{VU_r(t) + Q(t)\mu(t) + (P(t) - \kappa)\rho(t)|\phi(t)\} + \mathbb{E}\{Q(t)\chi(t) - V\alpha U_d(t)|\phi(t)\}; \quad (14)$$

$$\Phi_2(t) = \mathbb{E}\{(\kappa - P(t))\varepsilon(t)|\phi(t)\}.$$

The proposed algorithm. Our proposed algorithm is to minimize Eq. (13) by maximizing $\Phi_1(t)$ and $\Phi_2(t)$. The algorithm consists of two procedures: (1) Content Selection to maximize Φ_1, and (2) Energy Control to maximize Φ_2. Algorithm 1 presents the algorithm in details.

Algorithm 1 Social Content Scheduling Algorithm

1) **Content Selection**. In time slot t, from a set of social contents $f(t)$ fetched to the proxy so far, the proxy selects a set $r(t)$ to deliver to the mobile user and a set $d(t)$ to drop such that the following objective is satisfied:

$$\text{max:} \sum_{c\in r(t);r(t)\in f(t)} \Phi_1^r(c,t) + \sum_{c\in d(t),d(t)\in f(t)} \Phi_1^d(c,t)$$

$$\text{st:} \sum_{c\in r(t);r(t)\in f(t)} s_c \le \theta(t); \quad (15)$$

$$\sum_{c\in r(t);r(t)\in f(t)} \rho_c(t) \le \min(e(t) - e_t, P(t)), \text{where}$$

$$\Phi_1^r(c,t) = VU_c(t) + Q(t)s_c + (P(t) - \kappa)\rho_c(t);$$
$$\Phi_1^d(c,t) = Q(t)s_c - V\alpha U_c(t). \quad (16)$$

2) **Energy Control**. In time slot t, the proxy adds $\varepsilon(t)$ to $P(t)$ if $P(t) \le \kappa$.

The Content Selection procedure selects $r(t)$ and $d(t)$ to maximize the objective function in Eq. (15) with two constraints. The first constraint guarantees that the device's downlink data rate in time slot t is enough to deliver the selected contents, and the second constraint ensures the energy is sufficient for the mobile device to receive the contents. This problem is in fact a knapsack problem that can be solved by optimization tools, e.g., CPLEX [16]. The Energy Control procedure keeps track of the virtual queue $P(t)$, and only adds $\varepsilon(t)$ to $P(t)$ if $P(t)$ does not overreach κ yet.

C. Performance Analysis

We present several theory bounds of the proposed algorithm[3]. Let us denote U^* as the maximum long term utility of the problem in Eqs. (6a)–(6e) achieved by any stationary randomized algorithms. Notice that such an algorithm requires to know about future. Our proposed algorithm achieves a long term utility $\bar{U}$ arbitrarily close to U^* without knowing about future.

Theorem 1 (Utility lower bound): Our algorithm achieves a lower bound of the time average utility:

$$\bar{U} = \lim_{T\to\infty} \frac{1}{T} \sum_{\tau=0}^{T-1} \mathbb{E}\{U_r(\tau) - \alpha U_d(\tau)\} \ge U^* - \frac{\beta}{V}, \quad (17)$$

where β is a constant defined in Lemma 1.
The theorem shows that by choosing an arbitrarily large V, the achieved utility is arbitrarily close to the maximum value.

Theorem 2 (Real queue Q upper bound): Our algorithm restricts the storage limit for real queue Q as:

$$Q(t) \le V\alpha + \lambda_{max} \quad \forall t. \quad (18)$$

[3]The detailed proofs are omitted due to the space limitations.

The theorem indicates the real queue is bounded by $V\alpha+\lambda_{max}$ in any time slot. The larger V leads to a larger restriction on Q. That is, there is a trade-off $O(V, 1/V)$ for the size of the real queue and the achieved long term utility. That is, queue size increases by V leads to long term utility increase by $\frac{1}{V}$.

An interesting observation from this analysis is that we can control the storage size, which is not unlimited at the broker/proxy. Another interesting observation from this theorem is that we can control the time average queue delay. Queue delay is the time difference between a content is created and it is delivered to the mobile device. We define time average queue delay as follows: $\bar{D} = \frac{\bar{Q}}{\bar{r}} \le \frac{V\alpha+\lambda_{max}}{\bar{r}}$ where $\bar{r}$ is time average data rate from the broker/proxy to the mobile device. It is clear that queue delay can be controlled by V.

Theorem 3 (Energy upper bound): Our algorithm ensures that the energy consumed in any time slot is bounded by:

$$P(t) \le \kappa + \varepsilon_{max} \quad \forall t. \quad (19)$$

The theorem shows that our algorithm ensures energy budget at any time is not more than a threshold. We know that energy budget $P(t)$ shared for the prefetching system may increase when the device is charged. This theorem infers that the energy budget is controlled even when the device is charged fully for a long time.

D. A Near-Optimal Algorithm

We next present an efficient algorithm for content selection that runs in $O(nlog(n))$ with an approximation factor of 2. That is, the algorithm achieves $\Phi_1^{apx}(t) \ge \frac{1}{2}\Phi_1^{opt}(t)$, where $\Phi_1^{opt}(t)$ is the optimal solution for the problem in Eqs. (15)–(16) and $\Phi_1^{apx}(t)$ is the near-optimal solution.

Algorithm 2 A Near-Optimal Algorithm for Content Selection

Step 1: Given a set $f(t)$ of contents fetched by the proxy so far, select $r(t)$ to maximize:

$$\text{max:} \sum_{c\in r(t);r(t)\in f(t)} (\Phi_1^r(c,t) - [\Phi_1^d(c,t)]^+)$$

$$\text{st:} \sum_{c\in r(t);r(t)\in f(t)} s_c \le \vartheta(t). \quad (20)$$

This is done by:

1) For each $c \in f(t)$, calculate $w_c = \frac{\Phi_1^r(c,t) - [\Phi_1^d(c,t)]^+}{s_c}$.
2) Add c in $f(t)$ to $r(t)$ in decreasing order of w_c if $w_c > 0$ until $\sum_{c\in r(t)} s_c \le \vartheta(t)$ does not hold.

Step 2: Select $d(t)$ from the set $f(t) - r(t)$ by iterating through each content c in $f(t) - r(t)$, and add c to $d(t)$ if $\Phi_1^d(c,t) > 0$.

We first convert the problem in Eqs. (15)–(16) to a new problem. The energy consumption for receiving contents is linear to data size [14], i.e., $\rho(t) = \rho_a(t)s(t) + \rho_b(t)$, where $s(t)$ is the total amount of contents transferred in time slot t, and $\rho_a(t)$ and $\rho_b(t)$ are constants in each time slot, which are functions of the network type and condition. Therefore, the second constraint of the problem in Eqs. (15)–(16) becomes $\sum_{c\in r(t)} s_c \le \frac{\min(e(t)-e_t, P(t)) - \rho_b(t)}{\rho_a(t)}$, which leads to the relaxation of one of the two constraints. Let us define $\vartheta(t) = \min(\theta(t), \frac{(e(t)-e_t, P(t)) - \rho_b(t)}{\rho_a(t)})$. The problem in Eqs. (15)–(16)

is equivalent to the following problem:

$$\begin{aligned} \text{max:} \quad & \sum_{\substack{c\in r(t)\\ r(t)\in f(t)}} (\Phi_1^r(c,t) - [\Phi_1^d(c,t)]^+) + \sum_{c\in f(t)} [\Phi_1^d(c,t)]^+; \\ \text{st:} \quad & \sum_{c\in r(t); r(t)\in f(t)} s_c \le \vartheta(t). \end{aligned} \tag{21}$$

Notice that in the new objective function $\sum_{c\in f(t)}[\Phi_1^d(c,t)]^+$ is independent to $r(t)$. This allows us to select the contents for delivering $r(t)$ first, and then derive the contents for dropping $d(t)$. We solve the problem in Eq. (21) in two steps, as presented in Algorithm 2. The following theorems analyze the performance of the near-optimal algorithm.

Lemma 2: Using Algorithm 2 for Content Selection achieves a bound on $\Phi_1(t)$:

$$\Phi_1^{apx}(t) \ge \frac{1}{2}\Phi_1^{opt}(t) \tag{22}$$

Theorem 4 (Utility bound): By using Algorithm 2 for Content Selection, the system achieves the following bound on the long term utility:

$$\bar{U}_{apx} \ge \frac{1}{2}U^* - \frac{\beta_{apx}}{V}. \tag{23}$$

We note that using Algorithm 2 achieves the same bounds on the real queue and virtual queue as presented in Theorems 2 and 3.

IV. Performance Evaluation

A. Prototype System and Evaluation

Table I
Energy consumption (J) at devices for offline access to Facebook in 30 hours

Approach	WiFi	Cellular
Without Broker/Proxy	970.9 (6.9X)	16,783.4 (9.1X)
With Broker/Proxy	140.2	1,843.1

We illustrate the feasibility of our system by developing a prototype consisting of a broker/proxy program and an Android app, called Offline Facebook [17]. This app can download updates on users' News Feeds page at Facebook including the feed contents (texts, images, and videos), comments and likes, and provides a friendly GUI for the users to view as well as interact (e.g., comments and likes) with the downloaded contents offline without network connectivity (an example is presented in Fig. 3). The app can also receive updates from the broker/proxy program.

To quantify the benefit of the broker/proxy solution, we conducted the following experiments: We install the broker/proxy on an external Ubuntu machine, and the app on two Nexus One phones running Android 2.3.6. We configure the apps on the two phones to run with and without the broker/proxy, respectively. Without the broker/proxy, the phone checks with Facebook servers for the latest updates once every 5 minutes; in the other case, the broker/proxy performs the checking of updates and signals the phone using Google Cloud Messaging, if any relevant updates appear. Upon being signaled the phone establishes a connection to the broker/proxy and receives the updates. We consider two access networks: WiFi and cellular (T-Mobile), and use PowerTutor [18] to measure the power consumption of our app. We run 30-hr experiments using the same Facebook account, and present the results in Table I. This table shows that using the broker/proxy saves significant energy–we notice a 6.9-fold reduction with WiFi and a 9.1-fold reduction with cellular networks. The experimental results demonstrate the value of even a simplistic broker/proxy solution.

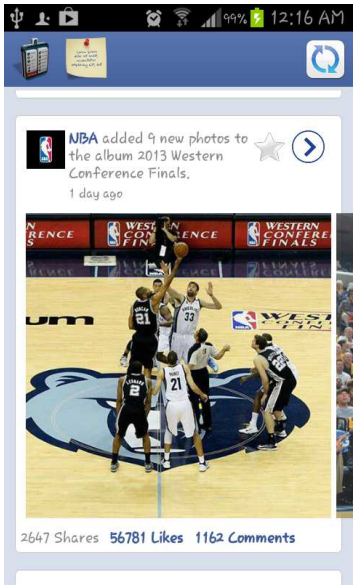

Figure 3. GUI for our data collection Android app [17].

B. Trace-driven Simulations

1) Dataset: To evaluate the performance of our proposed algorithm, we conducted trace-driven simulations using Facebook user data collected through our Android app [17]. We release our app to 10 users in America, Asia and Europe, and collect their traces between May 15 and 24, 2013. With their consensus, we log and collected information on their social attributes (e.g., Facebook friends and groups), news feeds attributes (e.g., their authors, created time, downloaded time, and size), device resources (e.g., battery and network conditions), user-app interactions (e.g., timestamps of clicks to view, comment, and like a news feed), and etc. To protect users' privacy, all Facebook objects, such as users, friends, groups and news feeds, are one-way encoded to prevent being traced back to the original users and contents. Our dataset includes 12,596 multimedia contents for all 10 users. The maximum number of contents for a user is 3,919, and the minimum number is 130. A user has 310 friends on average. The maximum number of friends is 503 and the minimum is 75. Each user clicks to view contents 693 times on average. The maximum ratio between the number of clicks and the number of contents is 95%, and the minimum ratio is 17%.

2) Settings: We built a Java based simulator that is driven by the collected traces: the feed stream trace was used to model Facebook's content arrival while the network trace to model network conditions for the simulation. Energy consumption was simulated using PowerTutor's models [18]. Optimization solver [16] was used to solve the optimization problem in the Content Selection procedure in Algorithm 1. We run the

simulations on a Windows 7 machine with an Intel Dual Core 1.8 GHz CPU with 4 GB RAM. We compare our algorithms, OPT (the optimal solution presented in Section III-B) and NOPT (the near optimal solution presented in Section III-D) with 3 baselines algorithms:

- *Rand*: This algorithm selects contents randomly to deliver to user.
- *Size*: Contents with smaller size are given higher priority for delivery selection.
- *Prob*: Contents with higher viewing probability are given higher priority for delivery selection.

In all algorithms, we do not download contents when the current energy level on the mobile device is lower than the energy threshold. The energy threshold in the simulation is set to the average energy level per user by default. For our algorithms, we set parameter $\alpha = 10$ to place a serious consideration on content dropping. κ is set to 20% of the maximum battery capacity. All users had been set to have the same energy budget periods in a day; more specially, a day time was divided into 3 periods: working period from 7 A.M. to 6 P.M, hangout period from 6 P.M. to 11 P.M., and home period 11 P.M. to 7 A.M. The corresponding usage energy percentage for these periods are 20%, 30% and 40%.

We consider several metrics to evaluate the algorithms.

- *Utility*: Accumulative viewing probability of all social contents received by a user.
- *Queue delay*: Average delay from the time a content is downloaded by the broker/proxy to the time it is delivered to a user.
- *Delivery ratio*: A ratio of the number of contents delivered to a user to the number of contents downloaded by the broker/proxy from Facebook for the user.
- *Hit late ratio*: The ratio of number of contents scheduled to be delivered to the user later than their actually viewing time to the total number of contents delivered to the user.
- *Effective energy*: Total energy consumed by the mobile device to download contents divided by the number of contents actually viewed by a user.

3) Simulation Results: We evaluate our system by showing the performance of our proposed algorithms under varying values of parameter V. This helps us understand the proposed algorithms' behaviors. Later, we compare our algorithms with other baseline algorithms.

Performance of the algorithms. We start with comparing OPT and NOPTs' performance under different V values. Fig. 4 presents the utility achieved by our algorithms. We observe that the increase of V leads to the increase of the utility in both algorithms. For example, at a very small value, $V = 10$, the utility is 51.52 for NOPT while at $V = 10^6$, the utility is 393.88. It is because at a larger V, the system becomes more hesitating to drop contents, and pushes content transfer to the users more strongly. A higher number of contents received by mobile device leads to a higher total viewing probability. This is supported by Fig. 5, which presents delivery ratio of contents delivered to the users. At $V = 10$, the delivery ratio is only 9%, but at $V = 10^6$, mobile device receives approximately 70%. This result proves the correctness of our performance analysis in Section III-C.

An interesting result in Figs. 4 and 5 is the two algorithms, OPT and NOPT, achieves very close utility and delivery ratio (our experiments also indicate they have very similar results for the other metrics). For example, at $V = 100$, NOPT achieves the utility of 79.14 while OPT has a slightly higher value, 80.94. The largest gap of utility between the two algorithms is only 4.36 in any V. The key difference between these algorithms is at the running time. Using optimization solver CPLEX to solve the OPT problems leads to the worst case running time of 2.4 secs in our experiments. The worst case running times for NOPT is only 78 ms. This indicates that NOPT runs much faster than OPT while achieving a very close performance. We thus suggest to use NOPT for the practical deployment purpose, and from now on, we do not report experiment results for OPT.

Although larger V leads to higher utility, it also leads to larger queue delay. For example, in Fig. 6 at $V = 10$, the queue delay for NOPT is 0.5 hours while at $V = 10^6$, the delay is more than 8 hours. A reason for the high delay is we set energy threshold to be high (average energy level). But we see a trend here. With larger V, more contents in queue are considered for being delivered to mobile devices. This observation is inline with our analysis: queue size and queueing delay are strongly dependent on V. Transferring a larger number of contents causes high queue delay. Note that in the social network context, queue delay in some sense indicates the satisfaction of user: how soon they can view social content updates from their friends.

We report effective energy according to V in Fig. 7. The lower the effective energy, the better the system works, because more energy is used for delivering right contents. Fig. 7 demonstrates that when V increases, the effective energy decreases and then increases. For example, at $V = 10^1$, the effective energy is 18 J per content; at $V = 10^4$, it decreases to 11 J per content; it increases back to 30 J per content when $V = 10^6$. This trend can be explained as follows. With a small V, queue size is dominated in making content selection for delivery and dropping, and viewing probability does not have much impact on decisions. Thus, the effective energy is low at small V. With a medium V, the viewing probability plays a larger role, and a higher number of contents clicked to view by the user is delivered, which contributes to achieving low effective energy. With a large V, the system delivers more contents than necessary if network and energy conditions permit. Therefore, there are more contents not clicked to view by the user delivered to the user. This leads to higher effective energy.

Compared with other strategies. Fig. 8 shows the effective energy to download contents viewed by users using each algorithm under two energy thresholds. For fairness, we set parameter $\alpha = 10000$ to prevent our algorithm from dropping contents so that all algorithms download all contents (with different schedules) under the same energy and bandwidth constraints in the simulations. The proposed algorithms exhibit better energy efficiency than the baseline algorithms: 15%–30% difference is observed.

Fig. 9 presents the hit late ratio of the downloaded contents, which indicates the timeliness of content downloads that are

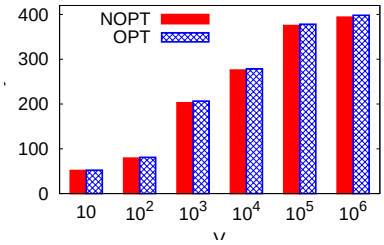

Figure 4. Utility.

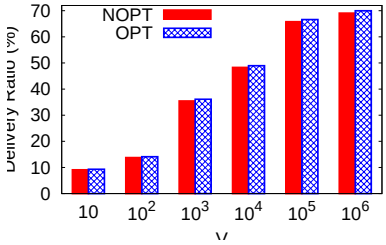

Figure 5. Delivery ratio.

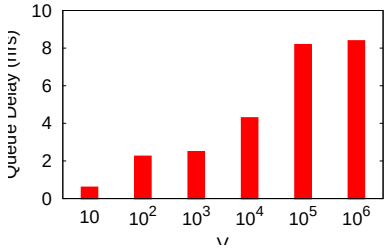

Figure 6. Queue delay.

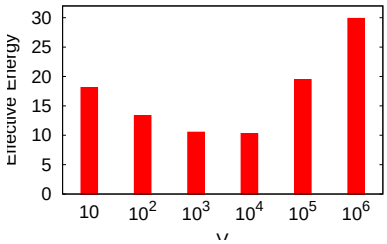

Figure 7. Effective energy.

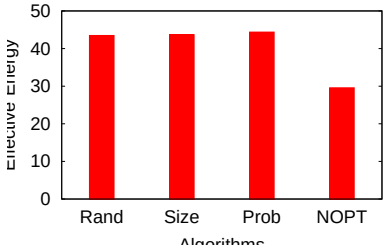

Figure 8. Effective energy.

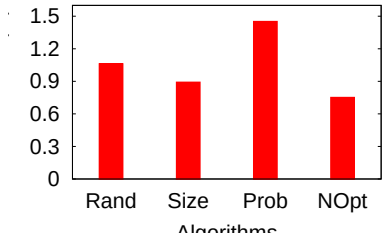

Figure 9. Hit late ratio.

viewed by users. We observe that our proposed algorithm achieves the best hit late ratio. Counter intuitively, the Prob algorithm performs worst in hit late ratio. This is because the algorithm fails to consider content size and resource conditions to download, although it prioritizes the contents on viewing probability. Downloading a long video may significantly delay the downloads of other smaller contents with similar viewing probability. In contrast, our algorithm jointly considers the viewing probability, content size, and resource conditions.

Table II
VIEWING PREDICTION ACCURACY (AUC) IN % OF EACH USER. THE AVERAGE ACCURACY IS 72%.

User	1	2	3	4	5	6	7	8	9	10
AUC	69	77	68	82	77	65	62	70	67	81

User-content relevance evaluator. We briefly present the results for our user-content relevance evaluator. We evaluate this component using a well-known metric AUC (the area under the ROC curve), which indicates the prediction accuracy. AUC varies from 0 to 100%. The closer to 100% AUC is, the higher accuracy the evaluator achieves. AUC is a good quality indicator for skewed distribution data set. We report AUC using 5-fold cross validation. More specifically, the data set is randomly divided into 5 partitions with equal size. 4 of 5 partitions were used in the training process, and the last partition was used to test learning model. The process was repeated 5 times, each using a different partition for testing. The AUC achieved by our evaluator varies from 62% to 81% with the average of 72%. Table II presents AUC in details for all considered users, which is fairly good: always $\geq 62\%$. Another observation is that the training overhead is low showing the efficiency of the proposed evaluation algorithm. The same Windows PC described above is used for the training process using MATLAB with 8 day data (i.e., including 4 partitions of the data traces). The average running time is only 0.006 seconds while the maximum is 0.138 seconds.

In summary, through real-world experiments and simulation studies, we have shown that:

- The broker-proxy approach is meaningful and yields significant performance benefits over a non-proxy based solution.
- The near-optimal solution is remarkably close in performance to the optimal solution that takes a long time to complete/converge.
- The near-optimal solution outperforms other state-of-the-art approaches.
- Carefully tuning the control parameter V significantly improves the performance.

V. RELATED WORK

Content prefetching has been studied in several domains, such as in processors that fetch data and instructions from RAM or disks in advance [19], and in distributed systems that fetch files from remote servers before receiving requests from applications [20]. Prefetching also attracts many attentions in mobile systems in which resources such as energy and network connectivity are constrained. Recent studies for prefetching in mobile systems can be classified into two categories: client-based [21]–[23] and broker/proxy-based approaches [24], [25].

In client-based approaches [21]–[23], mobile devices make prefetching decisions by themselves. Balasubramanian et al. [21] attempt to reduce cellular data traffic and support delay tolerant applications by exploiting intermittent WiFi connectivity. In [22], Schulman et al. predict time periods when cellular signal strength is strong, and transfer data in those periods to

reduce energy consumption, as low signal strength generally leads to higher energy consumption. Higgins et al. [23] propose to prefetch contents for applications in mobile devices based on the tradeoff between the energy cost and delay benefit.

In broker/proxy-based approaches [24], [25], mobile devices prefetch contents with supports from broker/proxy. In [24], Armstrong et al. employ a proxy to fetch updates for emails and weather reports, and batch the updates before sending them to mobile devices. Relevant updates are delivered based on interests expressed by users to the proxy. Mohan et al. [25] develop an ads delivery system, in which mobile devices fetch ads from servers via broker/proxies. They focus on the problem of prefetching ads to mobile devices before the ads are expired.

In contrast to the prior work, we develop an offline social network system for mobile devices to efficiently download *social media contents* leveraging *broker/proxy*, which is challenging for several reasons. First, social network sites typically do not offer notification services for new updates, as the number of updates is tremendous. Second, although there is relevance between users and contents, the arrival patterns of social media contents are hard to predict. In the current paper, we employ broker/proxy to optimally deliver social media contents to mobile users by carefully analyzing the user-content relevance and mobile device dynamics. Our system is built on fine-grained control algorithms that select what contents to send/drop with a strong emphasis on energy conservation. To our best of knowledge, our paper is the first of its kinds. The lessons learned in our work are crucial to app developers and social network providers for realizing high-quality mobile user experience for offline social network access.

VI. Concluding Remarks

Today, mobile access to current social media information is based on the connectivity of the end-user; our aim in this paper is to exploit the delay-tolerant nature of user access to social media streams to prefetch social media contents at a user's mobile device when connectivity is good and cheap. We develop and evaluate algorithms and an end-to-end system to make the mobile social media experience seamless, personalized, and cost-effective and efficient. The crux of our approach is to leverage more capable resources in the network to assist the mobile device in performing resource-intensive storage and computation tasks, even when the user is offline. Towards this end, we propose a broker-proxy architecture that can efficiently sort the social media contents based on their importance to the user; and schedule the content downloads to the mobile device based on content relevance and current system conditions. The feature is implemented via a mobile app on Android platform and used to collect social network traces. We then conducted experiments on our testbed and extensive trace-driven simulations to evaluate our proposed approach and system. Our evaluation results show the value of the distributed broker/proxy/device architecture and the associated algorithms that control the transfer of contents within this system and outperform the baseline approaches.

As indicated, a full fledged real world deployment of the proposed vision requires other issues that must be resolved. These include challenges introduced by scale (large number of mobile users), their wide distribution (global nature of social interactions), mobility (travelling users) and security (shared nature of wireless access). An interesting approach to address the scaling of mobile social network applications like the one described in this paper, is the utilization of the upcoming cloud computing infrastructure (public, private, and edge clouds) to implement brokers/proxies. Resource intensive techniques such as ranking of contents across users and across social media sites can execute in public clouds (data centers) or in regional proxies. A structured approach to address these issues with meaningful practical solutions are key topics being addressed in our future research.

References

[1] "Smart and social: Android phones in india," http://www.nielsen.com/us/en/newswire/2012/smart-and-social-android-phones-in-india.html, May 2012.

[2] "Always connected," https://fb-public.box.com/s/3iq5x6uwnqtq7ki4q8wk.

[3] R. Rahimi, N. Venkatasubramanian, and A. Vasilakos, "MuSIC: On mobility-aware optimal service allocation in mobile cloud computing," in *IEEE Cloud'13*.

[4] Y. Zhao, N. Do, S. Wang, C. Hsu, and N. Venkatasubramanian, "O2sm: Enabling efficient offline access to online social media and social networks," in *Proc. of ACM/IFIP/USENIX Middleware*, Beijing, China, December 2013, pp. 228–242.

[5] J. Ahn and et al., "Open user profiles for adaptive news systems: help or harm?" in *ACM WWW'07*.

[6] W. Chu and S. Park, "Integrating tags in a semantic content-based recommender," in *ACM WWW'09*.

[7] M. Gemmis and et al., "Personalized recommendation on dynamic content using predictive bilinear models," in *ACM RecSys'08*.

[8] F. Cacheda and et al., "Comparison of collaborative filtering algorithms: Limitations of current techniques and proposals for scalable, high-performance recommender systems," *ACM Transactions on the Web'11*.

[9] B. Sarwar and et al., "Item-based collaborative filtering recommendation algorithms," in *ACM WWW'11*.

[10] H. Ma, I. King, and M. Lyu, "Effective missing data prediction for collaborative filtering," in *ACM SIGIR'07*.

[11] D. Liu and et al., "Hybrid social media network," in *ACM MM'12*.

[12] J. Noel and et al., "New objective functions for social collaborative filtering," in *ACM WWW'12*.

[13] H. Ma and et al., "Recommender systems with social regularization," in *ACM WSDM'11*.

[14] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network applications," in *IMC'09*.

[15] L. Georgiadis, M. Neely, and L. Tassiulas, "Resource allocation and cross-layer control in wireless networks," *Foundations and Trends in Networking'06*.

[16] "IBM CPLEX optimizer," http://www-01.ibm.com/software/commerce/optimization/cplex-optimizer/.

[17] "oFacebook," http://www.ics.uci.edu/~dsm/oFacebook.

[18] "PowerTutor," http://ziyang.eecs.umich.edu/projects/powertutor.

[19] A. Smith, "Cache memories," *ACM Computing Surveys'82*.

[20] J. Kistler and M. Satyanarayanan, "Disconnected operation in the coda file system," *ACM Transactions on Computer Systems'12*.

[21] A. Balasubramanian, R. Mahajan, and A. Venkataramani, "Augmenting mobile 3G using WiFi," in *ACM Mobisys'10*.

[22] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, and C. Grunewald, "Bartendr: A practical approach to energy aware cellular data scheduling," in *ACM Mobicom'10*.

[23] B. Higgins, J. Flinn, T. Giuli, B. Noble, C. Peplin, and D. Watson, "Informed mobile prefetch," in *ACM Mobisys'12*.

[24] T. Armstrong, O. Trescases, C. Amza, and E. Lara, "Efficient and transparent dynamic content updates for mobile clients," in *ACM Mobisys'06*.

[25] P. Mohan, S. Nath, and O. Riva, "Prefetching mobile ads: Can advertising systems afford it?" in *ACM Eurosys'13*.