2014 43rd International Conference on Parallel Processing

NetMaster: Taming Energy Devourers on Smartphones

Yi Zhang
Department of Computer Science and Engineering
Hong Kong University of Science and Technology
Hong Kong SAR, China
yzhangbh@ust.hk

Yuan He, Xiaopei Wu, Yunhao Liu
School of Software and TNLIST
Tsinghua University
Beijing, China
{*he, xiaopei, yunhao*}*@greenorbs.com*

Wenbo He
School of Computer Science
McGill University
Montreal, Canada
wenbohe@cs.mcgill.ca

***Abstract*—Smartphones nowadays are installed with diverse applications, each of which consumes energy and bandwidth. As more and more applications are crowded into a smartphone, they cause serious problems with regard to battery life and bandwidth utilization. Existing proposals to tackle such challenges usually resort to two ways: avoiding energy-consuming network activities or improving communication efficiency in terms of power consumption. Those approaches either affect the smartphone users' experience, or offer little benefit in prolonging the battery life. Motivated by insightful understanding of users' habit, we in this paper propose a novel approach to orchestrate network activities of smartphone applications, based on user's habit. We implement our approach on smartphones as a middleware service called NetMaster. The performance evaluation with real traces shows that NetMaster reduces energy consumption of network activities by 77.8% in average and increases network bandwidth utilization by over 200%. The user experience is surprisingly well preserved. The chance of undesired interrupt during normal usage is less than 1%.**

***Keywords*-energy; mobile computing; optimization**

I. Introduction

The market of smartphone Apps (applications) dramatically grows in recent years. Smartphones nowadays are crowed with numerous diverse applications. Most of the applications (e.g. Facebook, weChat) require using the cellular network, which means considerable cost in battery power and network bandwidth. Hence, a natural but critical challenge on smartphone platforms is the conflict between the high demand of network usage by various Apps [1] and the constrained energy and bandwidth resources [2]. The amount of Apps on smartphones is ever growing, but for many years and in the foreseeable future, we do not see remarkable innovation in battery technology. Solving the above-mentioned challenge becomes a crucial and urgent task in the area of embedded systems and mobile computing.

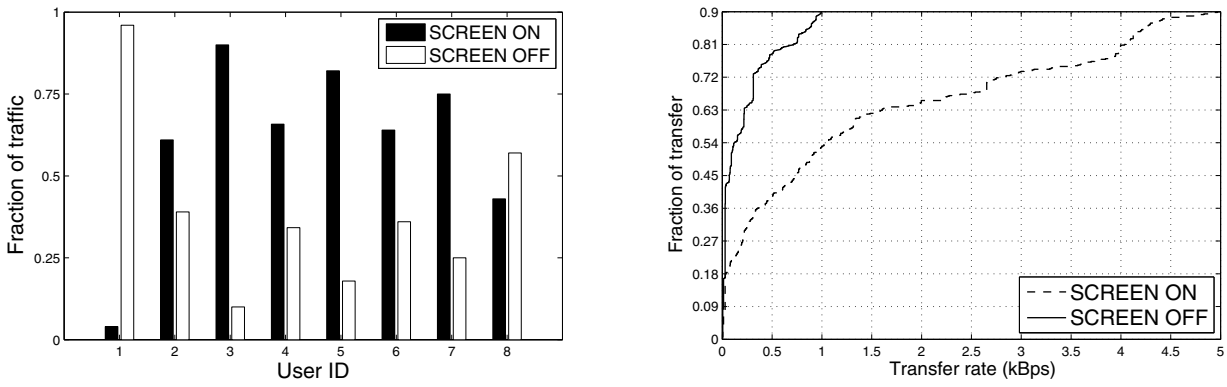
Various approaches have been proposed [3, 4, 5] to address the energy issue of smartphones, which usually resort to two ways: 1) improving communication efficiency in terms of power consumption [3, 4, 5, 7] or 2) avoiding energy-consuming network activities [2, 4, 9]. Nevertheless, both of them have apparent limitations: the first class of approaches indeed bring limited benefit [3, 5] with respect to energy. The applicable scenarios [4, 10] of improving communication efficiency are severely restricted as well. The second class of approaches [2, 11, 12] suffer a high risk of harming user experience. The fundamental problem behind this dilemma is the lack of in-depth understanding of smartphone users' behavior. The subtle tradeoffs among energy efficiency, user experience, and network utilization have great significance but often neglected by the existing works.

To address the above problems, we collect real traces from 8 users over 3 weeks for analysis. Further based on the results of existing measurement studies [9, 13, 14], we have two important findings: first, the screen-off network activities account for a significant portion of all network usage but have fairly flexible timing requirement. Second, the users' behavior of using smartphones exhibits highly regular patterns. Such interesting findings indicate great potential space to optimize energy efficiency of network activities under constrained bandwidth and simultaneously guarantee user experience .

In this paper, we propose a novel approach which predicts user behavior of network usage based on user habit mining. Accordingly, an online optimization algorithm is proposed to maximize energy saving and bandwidth utilization while also guarantee user experience. Our contributions are summarized as follows:

- We collect and analyze the real traces of 8 smartphone users for over 3 weeks, which reveal that smartphone users usually have strong habits in using smartphone Apps. We also measure the network traffic on smartphones at different time, so as to indicate a great potential to improve energy efficiency and network utilization.
- We propose a user habit oriented approach to predict smartphone usage and model the scheduling of network activities as a combination of multiple knapsack problems, which is proved to be NP-hard. We then propose a $\frac{1-\varepsilon}{2}$ approximate solution, which synthetically consider energy efficiency, network utilization, and user experience. To the best of our knowledge, we are the first to provide a scheduling approach of smartphone Apps with guaranteed performance with respect to energy efficiency.

0190-3918/14 $31.00 © 2014 IEEE
DOI 10.1109/ICPP.2014.39



(a) Network activity distribution

(b) Bandwidth utilization

Figure 1: Network activity profiling. In average, 40.98% of total network activities happen in screen-off state. For bandwidth utilization, 90% data transfer in the screen-off state are below 1kBps while 90% in the screen-on state are below 5kBps.

- We implement our proposal as a middleware service on smartphones called NetMaster. We evaluate NetMaster with real traces and compare its performance with the existing approaches. The results demonstrate the high efficiency and good generalizability of NetMaster. It reduces the average energy consumption of network activities by 77.8% and increases the network bandwidth utilization by over 200%, while surprisingly well preserves the user experience. The chance of undesired interrupt to normal usage is less than 1%.

The rest of paper is organized as follows: Section II describes the related work. Section III presents our motivation why we should eliminate the screen-off network activities. We present methodology design in Section IV. The system design and implementation of NetMaster is demonstrated in Section V. Section VI gives the system performance. We discuss the major concern in Section VII. Finally, we conclude our work in Section VIII.

II. Related Work

Network optimization receives increasing attention in recent years. Due to its complexity, researchers propose various methodologies focusing on different dimensions. We categorize them as follows.

Profiling. Several works have been proposed to characterize network activities in cellular networks. For example, the work in [3] profiles the usage patterns by tracking 255 smartphones. The work in [4] characterizes the relationship between users' application interests and their mobility properties. The work in [5] applies a passive measurement of mobile devices and shows that mobile traffic is dominated by multimedia content and mobile downloading applications. Overall, those works illustrate the diversity of user habits [6] and network activities [7]. However, they fail to propose efficient methods for optimizing energy consumption of network activities which will be addressed in our paper.

Data transfer optimization. To achieve energy saving at data transfer level, several efforts have been made. The work in [8] studies the impact of channel state on data transfer and reveals that good channel state can enhance energy efficiency of data transfer. Pathak et al. in [9] propose a comprehensive analysis of different methods on optimizing periodic data transfer. Huang et al. [2] characterize the screen-off data transfer using UMICH data set. They use batch and fast dormancy for off-line analysis. Although being effective, there still exist a large gap between off-line analysis and online optimization due to limited knowledge of user habits.

III. Motivation

Usually, an App falls into dormancy when we push "POWER" button to shut down the screen. But it can still remain normal activities in the background. To obtain a deep understanding of this mechanism, we collect real traces of 8 smartphone users for over 3 weeks. The users have sufficient diversity, at the ages of 20-30 and with different professions. By analyzing the traces, we find they involve different background network activities at the screen-off state. Moreover, as depicted in Fig. 1(a), network activities at the screen-off state accounts for 40.98% of all the activities. As stated in [2], this portion of network activities is low responsive and can be optimized aggressively. On the other hand, we find the radio utilization at the screen-on state is low as well. The radio utilization ratio refers to the percentage of screen-on time with active network communication. As illustrated in Fig. 2, the average radio utilization ratio is only 45.14%. It indicates that over 50% of radio-on time is wasted without any communication activities, though the screen is on. In addition, as shown in Fig. 1(b), the bandwidth utilization of 90% data transfer at the screen-off state is lower than 1kBps while 90% at the screen-on state is lower than 5kBps. This is far lower than the bandwidth provided by the carriers. The above mentioned findings clearly reveal great potential

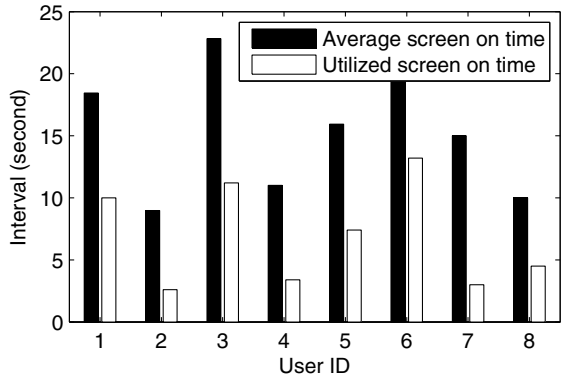


Figure 2: Screen-on time utilization profiling. By our analysis, the average radio utilization ratio in screen-on state is 45.14%.

space to improve radio utilization ratio at the screen-on state and to eliminate screen-off network activities for energy conservation.

The existing proposals to address the above issues, however, have limitations with different regards. The works in [2, 10] conduct off-line optimization by using simple methods (e.g. batch, delay) to aggregate or delay data transmissions. Intervals of 180s [10] and 100s [2] are adopted, respectively. Nevertheless, according to the user traces we collect, 17% user interactions fall just between two adjacent screen-off slots with intervals below 100 seconds. It reveals that the normal usage suffers a risk of being interrupted by those "interval-fixed delay" methods, because the user's habit is not appropriately taken into account.

Understanding user habit is of great significance for a solution against screen-off network activities. We conduct 3-week experiments with 8 users for user habit analysis and get two key observations. First, different users have rather distinctive usage patterns. To analyze the difference between two usage patterns, we introduce the Pearson parameter. Pearson parameter is used to estimate the correlation of two vectors with the same dimensions. It is calculated by:

$$\rho_{X,Y} = \frac{\sum(X-\overline{X})(Y-\overline{Y})}{\sqrt{\sum(X-\overline{X})^2 \sum(Y-\overline{Y})^2}} \quad (1)$$

In this equation, X and Y represent two usage vectors. Each vector have 24 dimensions representing the usage intensity in 24 hours. The intensity refers to the total times of usage in an hour. If the result is large, then X and Y are highly correlated which means two users share similar usage patterns. As depicted in Fig. 3, the average Pearson parameter of all users is 0.1353 which shows low correlation. This implies a one-fit-all method, e.g. delay or batch with fixed intervals [10][2], cannot be found for all users. Second, the Pearson parameter of the same user is much higher. By analysis, the average

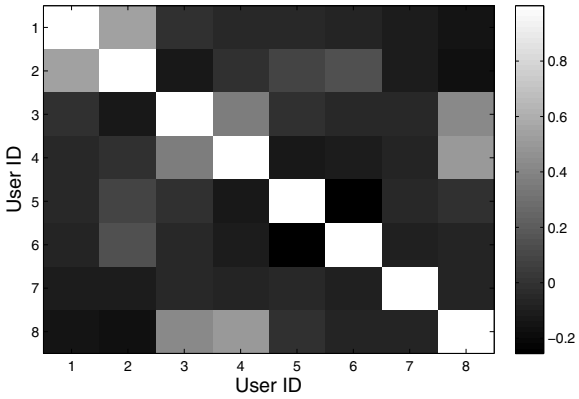


Figure 3: Pearson parameters for all users. (Avg=0.1353)

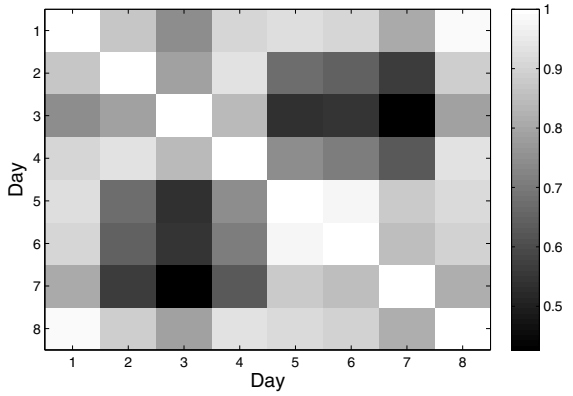


Figure 4: Pearson parameter for one user, ID=4 (Avg=0.8171)

Pearson parameter of each user is 0.54, including days with scattered user behaviours, which indicates a much higher correlation. For example, as illustrated in Fig. 4, the average Pearson parameter of user 4 is 0.8171 including the days with low values, e.g. the value between Day 3 and Day 7 is 0.464. This implies that for a single user, its usage pattern follows a regular daily pattern and is thus highly predictable. Overall, the above mentioned observations motivate us to develop a dynamic optimization scheme based on user habit prediction.

IV. Methodology Design

In this section, we discuss the methodology to eliminate screen-off network activities for saving energy online while also guarantee normal usage will not be interrupted. To realize this goal, the methodology contains three steps: predicting when user will use the smartphone; predicting when screen-off network activities will happen; determining the scheduling scheme to maximize the total energy saving while minimize the probability of interrupting users. More specific definitions will be discussed as follows.

Table I: Notations

Parameter	Description
t_i	time slot i
n_j	network activity in time slot t_j
$u(t_i)$	using phone in time slot t_i
$u(t_i)_j$	using phone in time slot t_i of day j in history. $u(t_i)_j = \{0, 1\}$
$n(p_m, t_i)$	network activity of App m in time slot t_i
$n(p_m, t_i)_j$	network activity of App m in time slot t_i of day j in history. $n(p_m, t_i)_j = \{0, 1\}$
$V(n)$	data size of the network activity n
$Pr[n(t_i)]$	probability of using network in time slot t_i
$Pr[u(t_i)]$	probability of using phone in time slot t_i
$thr(u)$	probability threshold to judge whether a time slot will contain user interaction
e_t	average energy consumption ratio
ΔE	energy saved by scheduling scheme
ΔP	penalty caused by scheduling scheme
$C(t_i)$	the capacity of slot t_i

A. Problem formulation

To formulate this problem, we will follow the previously mentioned steps.

Step 1: Predicting user active slots. User active slots mainly refer to the time slots of using the phone, i.e. the screen is on and the keyboard is unlocked. Assuming we have k days records, we define a time slot t_i to be a user active slot if it satisfies:

$$Pr[u(t_i)] = \frac{\sum_{j=1}^{k} u(t_i)_j}{k} \geq thr(u) \tag{2}$$

in which $thr(u)$ is the probability threshold for user active slot detection. We should mention that t_i doesn't have a fixed length. For all time slots satisfying equation (2), we merge them into a set U called user active slot set, in which $U = \{\forall t_i | Pr[u(t_i)] = \sum_{j=1}^{k} u(t_i)_j / k \geq thr(u)\}$.

Step 2: Predicting screen-off network active slots. Screen-off network active slots refer to the time slots when screen is off but there is still data transmission over mobile network. Similarly, we define a time slot t_i to be a screen-off network active slot if it satisfies:

$$Pr[n(t_i)] = \frac{\sum_{\forall m} \sum_{j=1}^{k} n(p_m, t_i)_j}{mk} \geq 0, (t_i \notin U) \tag{3}$$

Then, for all time slots satisfying equation (3), we merge them into a set T_n called screen-off network active slot set. For each time slot $t_i \in T_n$, we use n_i to denote the network activity happening in t_i and also add it into T_n. For each n_i, we use the sum of all data transferred/received in t_i to denote $V(n_i)$.

Step 3: Determining scheduling scheme. After getting U and T_n, the next step is finding the scheduling scheme to incorporate n_j into U. It should maximize saved energy, ΔE while minimize the penalty ΔP.

The saved energy mainly comes from reduced radio-on time by eliminating screen-off network activities. For the network activity n_j in T_n, ΔE_j from scheduling n_j is defined by $\Delta E_j = g(t_j)$ in which t_j is the slot n_j belongs to. Function g denotes the power model for the radio-on time. We use the results from [11] and [9] for g function.

For penalty, it denotes the probability of interrupting user interactions. To scale with ΔE, we introduce a scaling factor e_t to transform interrupting probability into energy [12]. Assuming we need to schedule a network activity n_j from $t_j \in T_n$ to $t_m \in U$, ΔP_j is defined by

$$\Delta P_j = \int_{t_j}^{t_m} e_t dt \int_{t_j}^{t_m} Pr[u(t)]dt \tag{4}$$

in which $\int_{t_j}^{t_m} Pr[u(t)]dt$ denotes the probability of using phone during the interval from t_j to t_m. Typically, if the scheduling schemes of n_i and n_j have overlapped slots, ΔP for overlapped parts will be calculated only once.

In addition, for each t_i in U, its capacity is defined by:

$$C(t_i) = Bandwidth \cdot t_i \tag{5}$$

in which $Bandwidth$ is the average bandwidth provided by the carrier. Based on the above mentioned definitions, the scheduling scheme is straightforward: given U and T_n, we want to find a scheduling scheme $S = \{s_{t_i}, ..., s_{t_m}\}$ which maximizes net saved energy $\Delta E - \Delta P$. For each $s_{t_i} \in S$, it contains network activities from T_n which are scheduled into $t_i \in U$. The optimization formula can be expressed as follows:

$$\begin{aligned} &Max \sum_{\forall n_j} (\Delta E_j - \Delta P_j) \\ &s.t. \sum_{j=1}^{k} V(n_j) \leq C(t_i), \forall n_j \in s_{t_i} \end{aligned} \tag{6}$$

To solve this problem, we transform it into a multiple knapsack problem. For each t_i, it denotes a *knapsack*. $C(t_i)$ is the *capacity* of *knapsack* t_i. In addition, each network activity n_j denotes an *item* and for each *item*, its *profit* is defined by $\Delta E_j - \Delta P_j$ and its *weight* is $V(n_j)$. The scheduling scheme is the most profitable *packing scheme* given *capacity* constraints. However, our problem is much harder because for any two adjacent user active slots, they share an overlapped itemset. In addition, for each item in one itemset, its profit is not independent because its ΔP might be overlapped with others. Therefore, we first refer to the solution under optimal condition to shed light on the original problem.

B. Solution under optimal condition

Assuming that we can obtain the user active slot set and the screen-off network active slot set accurately, then the scheme is to schedule every $n_j \in T_n$ into adjacent $t_i \in U$. By that means, we can get the maximized *profit* without incurring any ΔP. Furthermore, to solve the "overlapped itemset" problem, we first duplicate each n_j to create

independent itemset for each $t_i \in U$. Under this setting, the original problem has been reduced to the combination of single knapsack problem.

The single knapsack problem is known as an NP-hard problem and well studied by researchers. Due to the limitation of the article, we will omit NP-hard prove here. Based on dynamic programming, Ibarra et al. in [13] propose a fully polynomial approximation algorithm. This algorithm, denoted by *SinKnap*, can give us a $(1-\varepsilon)-approximation$ and the detailed proof can be found in [13]. Inspired by their work, we design a $\frac{1-\varepsilon}{2} - approximation$ algorithm ($\varepsilon \in [0,1]$) for our problem. It mainly contains four steps, as illustrated in Algorithm 1:

1) **Duplication**. For each n_i in T_n, we duplicate it.
2) **Sorting**. Sort the network activities of each user active slot according to their profit-to-weight ratios in the nonincreasing order.
3) **Dynamic programming**. Apply *SinKnap*[13] to find the best heuristic solution based on ε. It returns packing scheme for each $s_{t_i} \in S$.
4) **Filtering**. For each $n_j \in T_n$, check whether it appears twice in S. If n_j appears twice, choosing $s_{t_i} \in S$ with smaller $C(t_i) - V(n_j)$ and delete the other one. After filtering all duplicated ones, applying a greedy algorithm to add items for each $s_{t_i} \in S$ if possible.

Lemma IV.1. *The algorithm guarantees a $\frac{1-\varepsilon}{2}$ approximation ratio for proposed multiple knapsack problem with overlapped itemsets.*

Proof: Firstly, since all network activities are duplicated, we define $OPT_{dup}(t_i)$ as the optimal solution under this condition for slot $t_i \in U$ and $OPT(t_i)$ under the original condition. Since we may merge more network activities under duplicated condition into t_i, $OPT_{dup}(t_i) \geq OPT(t_i)$.

Based on *SinKnap*[13], let s_{t_i} denote the heuristic solution for slot t_i. It can guarantee that given any $\varepsilon < 1$, the s_{t_i} satisfies the following equation: $\frac{OPT(t_i)-s_{t_i}}{OPT(t_i)} < \varepsilon$ which is also equivalent to s_{t_i} is a $(1-\varepsilon)$-approximate solution.

Getting the $(1-\varepsilon)$-approximate solution for single knapsack problem, we need to filter out those duplicated network activities. After this step, we guarantee that every $n_{j*} \in S$ only appears once. Therefore, for any knapsack t_i, we have:

$$\sum_{\forall t_i} \sum_{\forall n_{j*} \in t_i} \Delta E_{j*} \geq \frac{1}{2} \sum_{\forall t_i} \sum_{\forall n_j \in t_i} \Delta E_j \quad (7)$$

$$\geq \frac{1-\varepsilon}{2} \sum_{\forall t_i} OPT_{dup}(t_i) \quad (8)$$

$$\geq \frac{1-\varepsilon}{2} \sum_{\forall t_i} OPT(t_i) \quad (9)$$

$$\geq \frac{1-\varepsilon}{2} OPT \quad (10)$$

where OPT is the optimal solution for total sets. Therefore, given $(1-\varepsilon)$ approximation for single knapsack subproblem, we get a $\frac{1-\varepsilon}{2}$-approximate solution for multiple knapsack problem with overlapped itemset as desired. Proved. ■

In addition, after filtering out duplicated ones, there also exist probability to add new network activities. So we adopt choosing scheme based on residual capacity to maximize the probability of adding new ones. Since duplicated n_j gives same profit in two slot t_i and t_{i+1}, we choose the slot with less residual capacity $C(t_i) - V(n_j)$. This provides more potential probability to merge more network activities. However, no matter what scheme we choose, the solution is constrained by:

$$\varepsilon OPT \leq OPT - \sum_{\forall t_i} \sum_{\forall n_{j*} \in t_i} \Delta E_{j*} \leq \frac{1+\varepsilon}{2} OPT \quad (11)$$

Therefore, the performance of "adding new ones" is lower bounded by εOPT in the worst case. So we apply a greedy algorithm for this part to control the computational overhead.

Algorithm 1: Algorithm for overlapped knapsack problem

Input: U for user active slot set, T_n for network active slot set
Output: S, scheduling scheme
$S = \emptyset$; **for** $j = 1; j < length(T_n); j++$ **do**
  for *every adjacent pair* $t_i, t_{i+1} \in U$ **do**
    if $t_i < n_j < t_{i+1}$ **then**
      $s_{t_i} = s_{t_i} \bigcap n_j$; *//Duplication*
      $s_{t_{i+1}} = s_{t_{i+1}} \bigcap n_j$;
for *each* $s_{t_i} \in S$ **do**
  Sort $n_j \in s_{t_i}$ according to: $\Delta E_j / V(n_j)$ in nonincreasing order; *//Sorting*
  $s_{t_i} = SinKnap(s_{t_i})$; *//Dynamic programming*
for *every* $n_j \in T_n$ **do**
  if $n_j \in s_{t_i}$ && $n_j \in s_{t_{i+1}}$ **then**
    if $C(t_i) - V(n_j) \geq C(t_{i+1}) - V(n_j)$ **then**
      Delete $n_j \in s_{t_i}$; **else**
        Delete $n_j \in s_{t_{i+1}}$; *//Filtering*
GreedyAdd(S);
return S;

C. Prediction based optimization

In the real-world environment, we cannot accurately know when and how long users will use smartphones. Hence, to schedule screen-off network activities, we should make predictions on both users and apps. However, making predictions faces two critical challenges.

First, fine-grained accuracy is hard to achieve. Based on our observation, users have nearly random usage patterns in the minute level, not to say the second level. On the contrary,

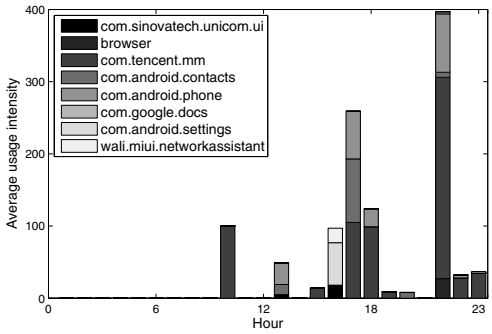


Figure 5: One week program pattern (ID=3). By analyzing the traces, we find only 8 (out of 23) apps have been used in one week and also have network activities.

the user habit is highly predictable at the hour level, as stated in Section III. Hence, we develop a light-weight hour-level prediction method to guarantee prediction accuracy.

Second, making predictions cannot guarantee 100% accuracy given randomness of user interactions. In addition, there also exist passive user interactions caused by accidental network activities which can not be predicted, e.g. new coming emails. This means eliminating screen-off network activities thoroughly may affect normal usage under this condition. Therefore, to handle those unpredictable incidents, we develop a real-time adjustment strategy.

1) Hour-level prediction method: To guarantee prediction accuracy, we propose a light-weight hour-level prediction method. We define "intensity" as the total times of usage in an hour. Usually, in some particular hours, the intensity stays nearly unchanged, e.g. there are near zero usage from 2am to 6am in our traces. While in others, there are intensity changes from day to day. Hence, our goal is to limit the impact caused by this change. As mentioned before, predicting user active slots needs $thr(u)$. To avoid interrupting normal usage, we develop a impact-based strategy of determining $thr(u)$. Basically, the threshold is defined as the max probability of interrupts, denoted as δ. Given the predicted user active slots, δ is the max value of $Pr(u)$ in the inactive slots. By carefully choosing δ, we can minimize the expected interrupt on users. Considering the different lifestyles during weekdays and weekends, we apply different δ strategies for them. In our experiments, we choose rather small δ to lower the probability of interrupting user activities, i.e. $\delta = 0.2$ for weekdays and $\delta = 0.1$ for weekends. However, making hour-level predictions also faces limitations. First, optimal δ is hard to choose. On one hand, large δ limits the number of slots in U which saves a large part of energy but suffers a high risk of interrupting normal usage. On the other hand, small δ can limit the probability of harming user experience but contributes little to energy saving. Second, finding optimal δ incurs large overhead. Since choosing optimal value requires a deep understanding of user's lifestyle. it needs weeks of experiments, large storage and sufficient computational ability which cannot be afford by mobile devices.

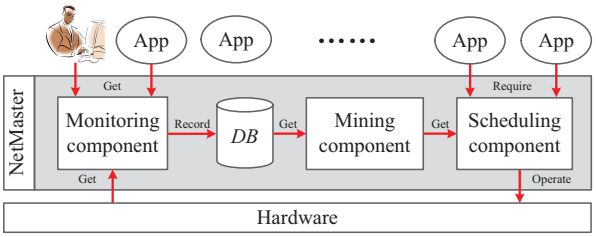


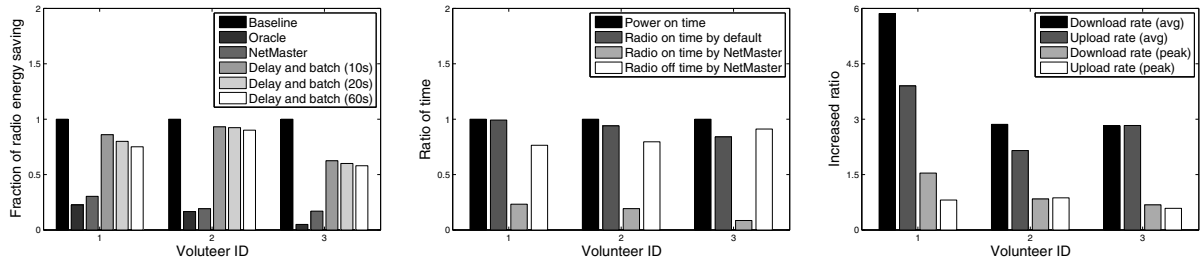
Figure 6: NetMaster Architecture. It has three components: monitoring component, mining component and scheduling component. A database (DB) is used to store user data.

Given the above limitations, we deem that hour-level prediction is also not sufficient for achieving satisfactory energy saving. Therefore, we develop a real-time adjustment strategy as a supplement.

2) Real-time adjustment strategy: The real-time adjustment strategy is responsible for powering on the radio when the screen is off or turning off the radio in the user active slots timely. To implement real-time control, we borrow the idea of duty cycle scheme in [14] with "Special Apps" as a supplement.

"Specific Apps" are the apps which have been used at least once along with network activities. For instance, as depicted in Fig. 5, only 8 (out of 23) apps are profiled as "Special Apps" for user 3. Moreover, *com.tencent.mm*, also known as weChat, is used for 669 times during one week and accounts for 59% of all usage. Therefore, by tracking activities of those apps, we can accurately detect user interactions and network activities while decreasing computational and storage overhead. When meeting a new installed app, we first recognize it as "Special Apps" to avoid making false operation. In addition, we also implement a duty cycle scheme in this strategy. This scheme is mainly used for imperfect predictions and accidental network activities which may cause user interactions. When the screen is off, this scheme will make the radio work in duty cycled manner: it wakes up the radio periodically to let "Special Apps" use the network. In addition, to save energy cost by falsely waking up the radio, we implement an exponential sleeping scheme. After setting the initial sleep interval T, when no user interactions and network activities are detected in the following wake-up period, the radio will sleep for $2T$, $4T$ and etc. Intuitively, we set this interval to be 30s in our experiements. By implementing this scheme, we can reduce energy cost of periodically waking up the radio while also guarantee user experience.

In words, the special cases handled by *Real-time adjustment strategy* are summarized as follows:



(a) Overall energy saving evaluation (b) Radio-on time evaluation (c) Bandwidth utilization improvement

Figure 7: Trace analysis

- **Usage outside the predicted slots.** If the user uses the smartphone outside the predicted slots, we'll check whether the foreground app is one of "Special Apps". If it is "true", then we will power on the radio. After usage, when no network activity is detected, the radio will be turned down by the duty cycle scheme.
- **Wasted radio-on slots.** When no network activity is detected, the duty cycle scheme takes control and implements the exponential sleep scheme.

V. System implementation

We implement the above-introduced scheme as a cross-App middleware, **NetMaster**, to schedule network activities. It works as a middle layer between applications and hardware, which mainly consists of three components: the monitoring component, the mining component and the scheduling component: **Monitoring component** records the information from users and Apps. The information is stored in the database on the smartphone. **Mining component** utilizes the database for hourly prediction. The prediction results will be broadcasted to the scheduling component. **Scheduling component** determines the scheduling strategy and executes it hereafter.

A. Monitoring component

There are four features that are recorded by the monitoring component: time, App, cellular network and screen.

We implement a hybrid model for accurate and energy-efficient recording. This model contains two triggers: event and time. The event trigger is used for catching changes of state variables, e.g. screen state. We register four broadcasters for each group and the corresponding receivers in NetMaster. When any state variable changes, the broadcasters will issue a message to the receiver.

To record non-state variables, e.g. received and transmitted bytes, we implement the time-triggered model. Considering different usage intensity in the screen-on state and the screen-off state, we set two timers for each of them. At the screen-on state, the user intensity is heavy, so a one-second timer is active. Otherwise, a longer timer set at 30 seconds is kept active in the screen-off state.

Moreover, frequently writing records to flash is energy-inefficient and slower than writing to memory [15]. To reduce energy and time cost, we use 500KB cache in memory to batch multiple writes together.

B. Mining component

This component makes hourly predictions for user active slots and network active slots at the screen-off state. It uses the prediction methods in Section IV and passes the results to the scheduling component.

C. Scheduling component

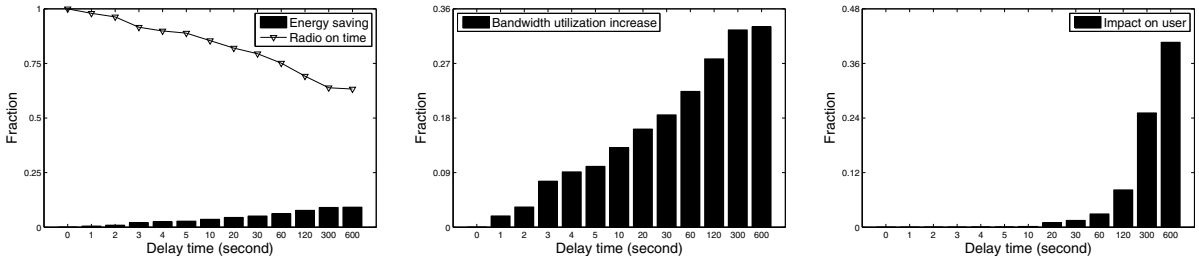
The scheduling component determines the scheduling strategy and executes real-time adjustment. It mainly consists of two parts: decision making and real-time adjustment.

1) **Decision making** This part manages activity distribution based on the previously mentioned algorithm in the optimal condition. The ϵ is set at 0.1 to guarantee good performance while control the computational overhead.
2) **Real-time adjustment** This part mainly controls the real-time network switch. This function is realized in a lower layer: we start a new children process and implement *svc data enable* or *svc data disable* to switch the radio on or off. The "Special Apps" can be easily got by querying the database. The duty cycle scheme is implemented as a background runtime service.

In addition, data transmission that spans across screen-on and screen-off states needs appropriate management. Typically, a transmission is not eliminated forcibly, considering some long-lasting network activities started by the user, e.g. stream video, Skype communication. We realize this function by polling system service of *TELEPHONY_SERVICE*. Only when no data transmission is detected, the duty cycle scheme will take over the control.

VI. Evaluation

We evaluate NetMaster on Android 4.1.1 and recruit three volunteers to conduct comparison experiments. The testing platforms are HTX One X, Lenovo A390T, and



(a) Delay impact on radio-on time (b) Delay impact on bandwidth utilization (c) Ratio of affected user activities

Figure 8: Off-line analysis on delay impact

Sharp 330T. We install NetMaster on each device and ask volunteers to freely use them without any restriction. We make this setting to allow the variability of user habits such that NetMaster is tested under different circumstances. The cellular network ISP is China Unicom, who provides WCDMA communications. We evaluate NetMaster in two dimensions: general performance and user experience. In addition, we also compare it with the approaches proposed in [10] and [2]. The impact of different parameter settings is examined as well.

A. General Performance

We demonstrate the performance of NetMaster with three metrics: energy savings, radio-on time and bandwidth utilization.

In the experiments, we adopt the power model proposed in [5, 8, 11] to estimate network energy consumption. To compute the energy saving, we implement two class experiments: with NetMaster and Without NetMaster. As mentioned in [11], energy consumed in network communication take the majority of total energy consumption. So we mainly focus on the reduced radio-on time on each device. For comparison, we also implement "naive delay and batch" proposed in [10] and [2], which uses a fixed interval to aggregate/delay screen-off network activities. To test the performance of different intervals, we use 10s, 20s and 60s as testing intervals, which do not affect user interactions. To obtain the ground truth, we apply off-line analysis to derive the optimal results for each volunteer. The optimal result refers to the minimal energy cost for the same network activities. As illustrated in Fig. 7(a), NetMaster can save 77.8% energy consumption of network activities in average. In 81.6% of all the tests, the gap between NetMaster and the optimal result is below 5%. The reason is that NetMaster not only eliminates screen-off network activities, but also turns off radio whenever necessary. This operation greatly reduces unnecessary radio-on time and increases bandwidth utilization.

The energy savings offered by the "naive delay and batch" scheme is 22.54% in average, which is apparently worse than the performance of NetMaster. We also notice that even in the worst case (e.g. volunteer 1), the gap between the optimal result and NetMaster is only 11.2%, which is far beyond 0.45 approximation factor defined by ϵ. Furthermore, we profile the performance in terms of radio-on time and bandwidth utilization improvement, as shown in Fig. 7(b) and Fig. 7(c). NetMaster saves 75.39% inefficient radio-on time in average. In addition, NetMaster achieves in average 3.84x for download rate while, 2.63x for upload rate. However, we also notice that NetMaster doesn't increase the peak rate. Note that the peak rate is determined by the channel state, no matter what scheduling scheme is used. We include this part in our future work.

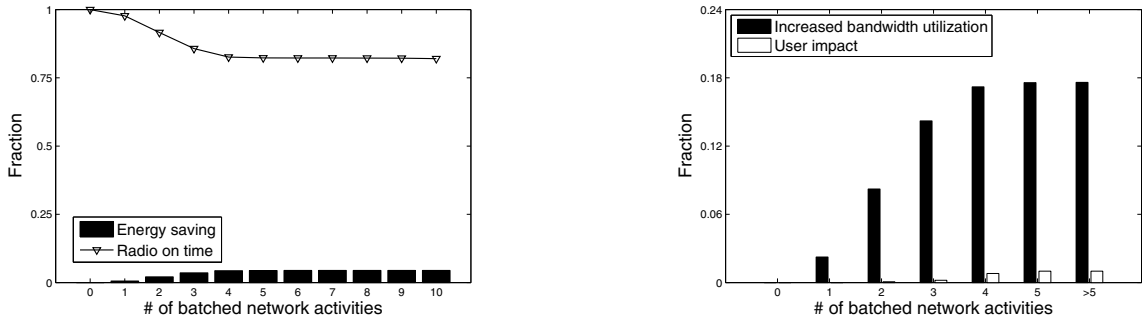
B. User Experience

User experience refers to whether NetMaster makes wrong decisions and blocks network when there are user interactions. In the experiments, we told the volunteers to turn on 3G network if the radio is turned off, as long as they need to access the network. Hence, we can track the appearance of *com.android.settings*, using the method *getMobileDataEnabled* as the indication of false operations. If this process appears foreground and the network switch incurred by the user is detected, we will record this activity sequence as the consequence of a "wrong decision". According to our traces, in all the 319 times of appearance of *com.android.settings*, only 1 wrong decision occurs. After careful examination, we find that the unique instance is due to the wrong return value of the method *getSystemService(TELEPHONY_SERVICE).getDataActvity*. That phenomenon can be eliminated by powering on 3G network before accessing the network. Generally, we can control the possibility of interrupting users under 1%.

C. Comparisons

We also compare the performance of NetMaster with the schemes of delay and batch, respectively. For each method, we test it under different parameter settings to obtain a comprehensive comparison with NetMaster.

Delay method puts off network activities for some intervals, during which the radio is shut down. In the experiments, we use different intervals varied from 1 second to 600



(a) Batch impact on radio-on time

(b) Batch impact on bandwidth & user

Figure 9: Off-line analysis on batch impact

seconds. We test the performance in four dimensions: radio-on time, bandwidth utilization, energy saving, and impact on user activity. The results are shown in Fig. 8. In Fig. 8(a) and Fig. 8(b), we can see that by increasing the delay interval, the radio-on time can be reduced by 36.7% (interval=600s), and the bandwidth utilization can be increased by 33.05%, but the energy consumption is only cut off by 9.2%. This is because it fails to avoid wasting radio-on time. In addition, we also examine the affected user activities which occur in the delay interval. By analyzing the traces, we notice that the ratio of interrupted user activities increase along with the delay interval. By applying 600 seconds delay interval, the ratio exceeds 40% which indicates normal usage is affected severely. On the contrary, applying small delay interval (e.g. 5s) gives little improvement on energy saving and bandwidth utilization increase. Therefore, the gap between interrupted usage and saving energy cannot be filled if using the delay method alone.

For the batch method, we test it by aggregating different maximum numbers of consecutive network activities. To compare it with NetMaster, we limit the probability of interrupting user activities to 1% or lower. As depicted in Fig. 9(a) and Fig. 9(b), the performance is enhanced with more aggregated network activities. The radio-on time can be eliminated by 17.7% while the bandwidth utilization can be increased by 17.6%. But its performance does not improve when the max number exceeds five. The reason may be that users do not usually start too many (larger than 5) network streams simultaneously, given the interrupting probability constraint.

Overall, neither delay or batch can achieve good performance alone. Compared with NetMaster, neither of them can guarantee sufficient energy saving and low probability of interrupting user interactions simultaneously.

D. On parameter settings

We investigate the impact of different parameter settings through trace analysis. According to their functions, we mainly focus on the impact of duty cycle and predicting threshold.

We compare different sleep intervals for exponential duty cycle scheme and the performances of other duty cycle schemes. As shown in Fig. 10(a), the increase of sleep intervals reduces the radio-on time significantly. In reality, the interval can be tuned to satisfy different user requirements. In Fig. 10(b), we can see the exponential scheme apparently outperforms the fixed and random sleep, which also contributes a lot to energy saving.

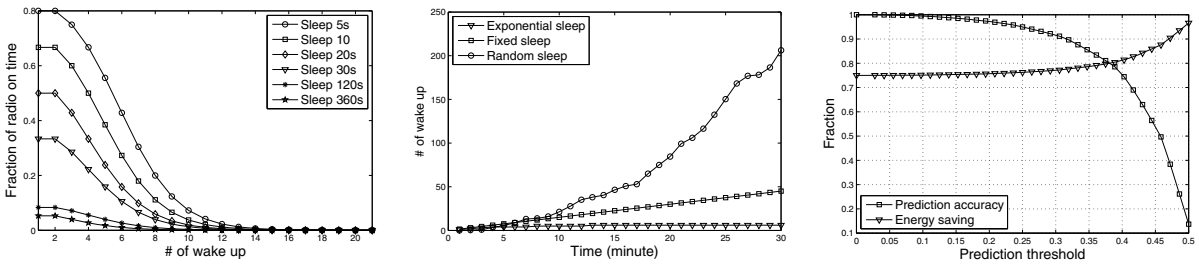
As for the impact of different prediction thresholds, the prediction accuracy is defined as the ratio of user activities falling inside the predicted user active slots and the energy saving is the ratio of the saved energy using tested threshold to the oracle value. As shown in Fig. 10(c), the value of δ for balancing between energy saving and prediction accuracy is 0.37. However, as stated in our design, preventing interrupting user activities is the first-place concern. Thus we choose a smaller value $\delta = 0.2$ for weekdays while $\delta = 0.1$ for weekends to eliminate the probability of interrupting users. Under this setting, we can guarantee the expected chance of undesired interrupt to normal usage is less than 1%.

VII. LIMITATIONS

Although conducting the comparison experiments for real smartphone users, the number of volunteers is rather small. We will recruit more volunteers to analyze their habits and estimate the performance of NetMaster in our future work.

Some measuring errors may be introduced in the experiments. In some recent works [3, 8, 11], they connect a power monitor to the smartphones to analyze its energy consumption. Although this method provides high accuracy, it is not applicable in measuring energy consumption in daily life. That is why we use a model-based approach, which is less accurate than a device-based approach. To improve the measuring accuracy, we will address the issue of measurement accuracy in our future work.

The hidden impact exists with NetMaster as well. The hidden impact refers to the potential risk of interrupting normal



(a) Radio-on time evaluation (b) Switch number evaluation (c) Prediction accuracy evaluation

Figure 10: Parameter analysis

usage. For example, in the user inactive slots, the duty cycle of radio is very low. Some push services, like Facebook, may be delayed and cause impact on user experiences. To understand and eliminate this hidden impact, we will put more efforts in analyzing user habits and App behaviours.

VIII. Conclusion

In this paper, we make insightful user habit analysis on real-traces from 8 users over 3 weeks. In addition, we propose a user habit oriented approach to predict smartphone usage and first model the scheduling of network activities as a combination of multiple knapsack problems. We derive a $\frac{1-\varepsilon}{2}$ approximate solution and implement it as a middleware service on smartphones called NetMaster. In real world experiments, it reduces the average energy consumption of network activities by 77.8% and increases the network bandwidth utilization by over 200%, while surprisingly well preserves the user experience. The chance of undesired interrupt to normal usage is less than 1%.

Acknowledgements

This work is supported in part by National Basic Research Program (973 program) under Grant of 2014CB347800, NSFC under Grants No.61170213 and 61373146, and NSFC Major Program under Grants No.61190110, and NSERC under Grant RGPIN 418521-12.

References

[1] X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: multi-layer profiling of android applications," in *Proceedings of ACM MobiCom*, 2012, pp. 137–148.

[2] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck, "Screen-off traffic characterization and optimization in 3g/4g networks," in *Proceedings of ACM IMC*, 2012, pp. 357–364.

[3] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proceedings of ACM MobiSys*, 2010, pp. 179–194.

[4] I. Trestian, S. Ranjan, A. Kuzmanovic, and A. Nucci, "Measuring serendipity: connecting people, locations and interests in a mobile 3g network," in *Proceedings of ACM IMC*, 2009, pp. 267–279.

[5] G. Maier, F. Schneider, and A. Feldmann, "A first look at mobile hand-held device traffic," in *Proceedings of Springer PAM*, 2010, pp. 161–170.

[6] X. Chen, X. Wu, X.-Y. Li, Y. He, and Y. Liu, "Privacy-preserving high-quality map generation with participatory sensing," in *Proceedings of IEEE INFOCOM*, 2014.

[7] 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 *Proceedings of ACM SIGMETRICS*, 2013, pp. 29–40.

[8] A. Schulman, V. Navda, R. Ramjee, N. Spring, P. Deshpande, C. Grunewald, K. Jain, and V. N. Padmanabhan, "Bartendr: a practical approach to energy-aware cellular data scheduling," in *Proceedings of ACM MobiCom*, 2010, pp. 85–96.

[9] A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," in *Proceedings of ACM EuroSys*, 2012, pp. 29–42.

[10] 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 *Proceedings of ACM WWW*, 2012, pp. 51–60.

[11] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of ACM MobiSys*, 2012, pp. 225–238.

[12] W. Zhang, Y. Wen, and D. O. Wu, "Energy-efficient scheduling policy for collaborative execution in mobile cloud computing," in *Proceedings of IEEE INFOCOM*, 2013, pp. 190–194.

[13] O. H. Ibarra and C. E. Kim, "Fast approximation algorithms for the knapsack and sum of subset problems," *Journal of the ACM*, vol. 22, no. 4, pp. 463–468, 1975.

[14] J. Polastre, J. Hill, and D. Culler, "Versatile low power media access for wireless sensor networks," in *Proceedings of ACM SenSys*, 2004, pp. 95–107.

[15] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang, and Q. Li, "Optimizing background email sync on smartphones," in *Proceeding of ACM MobiSys*, 2013, pp. 55–68.