A Personalized Network Activity-Aware Approach to Reducing Radio Energy Consumption of Smartphones

Yeseong Kim, Boyeong Jeon, and Jihong Kim, *Member, IEEE*

Abstract—The radio energy consumption takes a large portion of the total energy consumption in smartphones. However, a significant portion of radio energy is wasted in a special waiting interval, known as the *tail time* after a transmission is completed while waiting for a subsequent transmission. In order to reduce the wasted energy in the tail time, the fast dormancy feature allows a quick release of a radio connection in the tail time. For supporting the fast dormancy efficiently, it is important to accurately predict whether a subsequent transmission will occur in the tail time. In this paper, we show that there are strong personal characteristics on how user interacts with a radio network within the tail time. Based on these observations, we propose a novel personalized network activity-aware predictive dormancy technique, called Personalized Diapause (pD). By automatically identifying user-specific tail-time transmission characteristics for various network activities, our proposed technique takes advantages of personalized high-level network usage patterns in deciding when to release radio connections. Our experimental results using real network usage logs from 25 users show that pD can reduce the amount of the wasted tail time energy by 51 percent on average, thus saving the total radio energy consumption by 23 percent with less than 10 percent reconnection increase.

Index Terms—Smartphone, radio network, radio energy, tail time

1 Introduction

A significant portion of the total energy consumption in smartphones comes from the energy consumed in their radio device modules [1], [2], [13]. For example, in our energy measurement study for 3G smartphones, we have observed that radio communications are responsible for about 30 percent of the total energy consumption. With more smartphone apps interacting with remote data centers and servers, it is expected that the radio energy consumption in smartphones will be steadily increasing. Therefore, an efficient radio energy management is critical in achieving a high system-level energy efficiency of a smartphone.

In 3G and 4G network devices, a significant portion of the radio energy is wasted during a special waiting interval, known as *the tail time*. Once a radio connection is released, reconnecting to a mobile network again requires an extra overhead of allocating radio resource in the mobile network, resulting in a long delay to a mobile device and an extra signaling overhead to the mobile network. In order to avoid frequent radio reconnections, after a data transmission is completed, a radio connection is not immediately released. Instead, it is maintained during the tail time interval of a fixed-length T_{tail}, expecting that a subsequent transmission occurs in the tail time. However, if there is no transmission in the tail time, a large amount of the radio energy is wasted. For example, in our 3G radio power measurement study from 25 Android smartphone users, we observed that about 40 percent of the total radio energy consumption is wasted in the tail time while waiting for a subsequent transmission which did not actually occur in the tail time. Thus, saving wasted energy in the tail time is very important in achieving a high radio energy efficiency.

In order to save the wasted energy during the tail time, a new specification, called the *fast dormancy* feature [3], was proposed. The fast dormancy feature allows a smartphone radio module to quickly release its radio connection *even in the tail time*, thus reducing the wasted energy in the tail time when there is no subsequent transmission in the tail time. In order to utilize the fast dormancy feature efficiently, it is important to predict whether or when a subsequent transmission will occur in the tail time, after the current data transmission has been finished. For example, if the subsequent transmission is incorrectly predicted to occur in the tail time, the energy wasted in the tail time cannot be saved. On the other hand, if the subsequent transmission is incorrectly predicted not to occur in the tail time, a radio connection should be re-established with an additional radio resource allocation.

Existing predictive dormancy techniques such as [4], [5] are seriously limited in their applicability, missing many potential opportunities for exploiting the fast dormancy feature. For example, TOP [4] relies on explicit hints from apps on the subsequent transmission in deciding if a radio connection should be released or not after a data transmission is completed. RadioJockey [5], on the other hand, is more

- *Y. Kim is with the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA 92093. E-mail: ong@davinci.snu.ac.kr.*
- *B. Jeon is with SAP Labs, Kangnam-Ku, Seoul 135-270, Korea. E-mail: byjeon@davinci.snu.ac.kr.*
- *J. Kim is with the Department of Computer Science and Engineering, Seoul National University, 1 Gwanak-ro, Gwanak-gu, Seoul 151-742, Korea. E-mail: jihong@davinci.snu.ac.kr.*

Manuscript received 12 June 2013; revised 28 May 2014; accepted 23 Apr. 2015. Date of publication 11 May 2015; date of current version 2 Feb. 2016. For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below. Digital Object Identifier no. 10.1109/TMC.2015.2431712

general in that it does not require apps' explicit hints on future network transmission. However, RadioJockey is useful only for *non-interactive* tasks from *background apps* (such as updating daily weather forecast). Since popular apps are user-interactive apps, this limitation seriously restricts the applicability of RadioJockey.

Intuitively, what the existing techniques lack is a *systematic* and *automatic* way of extracting some *general-purpose* quantitative metric that can be used to predict the likelihood of the next network transmission in the tail time. For example, RadioJockey suggests a systematic and automatic solution but it is not general-purpose, being limited to non-interactive background apps only. In order to automatically estimate future network transmission patterns of apps without *a priori* knowledge on next transmission behavior, there are two key technical challenges. First, we should identify a right abstraction level on which information can be collected/profiled so that future radio network transmissions can be predicted. Second, we should show that the information gathered at the proposed abstraction level is indeed highly correlated to future radio network transmissions in the tail time.

In this paper, we argue that *a network activity* is a right abstraction level on which user's interaction patterns with a network should be monitored. Roughly speaking, a network activity can be considered as a semantically meaningful network-related action. For example, browsing a web page and fetching new emails are example network activities. Using a detailed user study, we show that network transmission patterns collected at the network activity level can be used in accurately predicting future tail-time transmissions. Furthermore, we observe that there are strong personal characteristics on how user interacts with a network in the tail time. In this paper, we present a novel network activity extraction technique that automatically classifies semantically equivalent network activities. Our technique, which exploits program contexts [6], partitions an app's network activities into a small number of equivalent network contexts.

By carefully monitoring how each user reacts at each network context, we can develop an efficient personalized predictive dormancy technique. In this paper, we propose a novel general-purpose personalized network activity-aware dormancy technique, called Personalized Diapause (pD in short). Unlike existing techniques, pD is based on *an automatic* and *systematic* approach that can be applicable to *any type* of apps. By automatically extracting various user-level network activities from a system software and identifying user-specific tail-time transmission characteristics for network activities, our proposed technique predicts future tail-time transmissions accurately, thus significantly reducing the amount of wasted energy in the tail time by exploiting the fast dormancy feature efficiently.

In order to evaluate the effectiveness of pD, pD was implemented on Android 2.3 (Gingerbread) smartphones. We have evaluated our proposed pD technique using real network usage logs from 25 users. Our experimental results show that pD can reduce the amount of energy wasted in the tail time by 51 percent on average while the number of radio reconnections is increased by less than 10 percent over when no fast dormancy feature is used. An energy-efficient tail time management by pD reduced the total radio energy consumption by 23 percent, resulting in 7 percent energy saving in the total energy consumption.

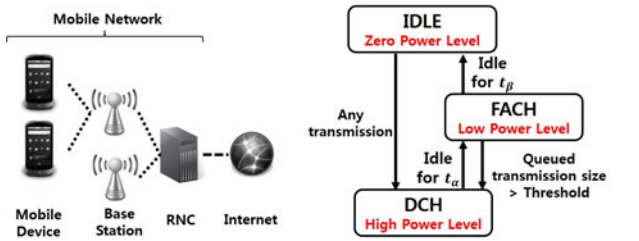


Fig. 1. 3G network management.

The design and implementation of the pD technique described in this paper is based on our earlier work [7]. We have improved the pD algorithm significantly over one presented in [7]. Furthermore, this paper presents a detailed quantitative analysis of a smartphone network usage study and a thorough evaluation of pD.

The rest of the paper is organized as follows. We briefly review an overview of 3G radio network management in Section 2. In Section 3, we explain key motivations of pD, emphasizing personalized tail-time network usage characteristics from our smartphone user study. We describe the proposed pD technique in detail in Section 4. Experimental results are presented in Section 5. Related work is discussed in Section 6 while Section 7 concludes with a summary and future work.

2 Background

2.1 Radio Network Management

As shown in Fig. 1a, a mobile device such as a smartphone is connected to Internet through a base station and a radio network controller (RNC) in 3G network systems. Since multiple mobile devices can be connected to a mobile network, the RNC manages limited radio resources such as transmission channels using the radio resource control (RRC) protocol which specifies how a mobile device changes its operating states. Since a radio device of a smartphone consumes different power depending on a RRC state, the radio energy consumption varies significantly by its RRC state controlled by the RNC [8], [9].

Fig. 1b shows a typical state machine of a 3G radio device. The 3G network state machine consists of three states, IDLE, dedicated channel (DCH) and forward access channel (FACH). Each edge in Fig. 1b represents a state transition and its trigger condition. For example, the state of a radio device remains at IDLE when the network is inactive, and the radio device consumes almost no power. Once there is a transmission to/from the radio device, the state transitions from IDLE to DCH in order to reserve a dedicated channel. At DCH, the radio device is maintained at a high power level. If there is no more transmission during a special waiting interval whose length is t_α, the state transitions to FACH in which the radio device shares a channel with other devices in the FACH state. The radio power in FACH remains at a low level. While at FACH, if additional transmissions occur at FACH and the size of queued transmissions exceeds a threshold of the radio link network, the state transitions back to DCH. In contrast, the radio device

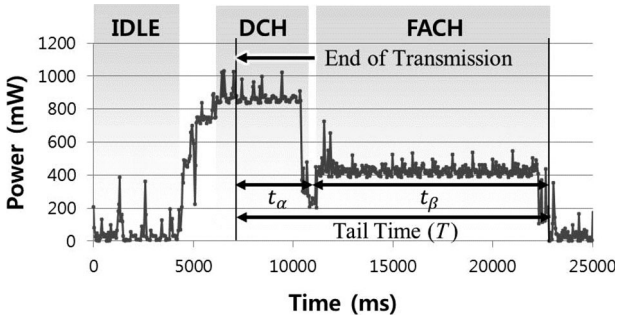


Fig. 2. 3G energy consumption.

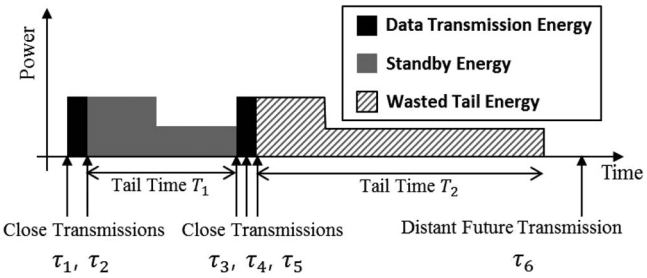


Fig. 3. Energy breakdown in a tail time interval.

returns back to IDLE when there is no transmission at FACH during another special waiting interval whose length is t_β.

Intuitively, if there is no subsequent transmission after a transmission is completed, a large portion of radio energy is wasted during two waiting intervals, collectively known as the tail time, whose length T_{tail} is $t_\alpha + t_\beta$. Since a radio resource allocation overhead incurs when a RRC state goes up from a low power state to a high power state, the tail time is used to avoid frequent such state transitions. For example, if a RRC state transitions from IDLE to DCH, a mobile device experiences a long delay, about 2.1 seconds, and the mobile network suffers from additional signaling overhead. Thus, in order to mitigate excessive resource allocation overhead, a radio device maintains its power state during the tail time interval, I_{tail}, expecting that a transmission is likely to occur in the tail time. However, if there is no transmission in I_{tail}, the mobile device wasted its radio energy consumed in the tail time.

In order to understand how a radio device actually consumes its radio energy during the tail time, we measured power consumption of smartphones using Agilent 34410A multimeter in a similar fashion to one used in [8]. The results are described in Fig. 2 as an example. In our local 3G measurements, t_α is 3 seconds and t_β is 12 seconds, thus the radio device waits for a subsequent transmission in the tail time, up to 15 seconds. If there is no transmission in the tail time, 7380 mJ is wasted.[1]

2.2 Fast Dormancy Feature

In order to reduce the wasted energy in the tail time, a new feature, called *fast dormancy*, was proposed in 3GPP [3]. By sending a control message to the RNC, a mobile device can immediately release its radio connection, thus saving energy otherwise wasted in the tail time. However, in spite of its high potential for saving the wasted radio energy in the tail time, the fast dormancy feature has not been widely utilized in 3G networks. For more recent 4G networks, where a large amount of energy is still wasted in the tail time, the fast dormancy support was not yet equipped. One of the main technical obstacles to employing the fast dormancy feature in mobile networks is that it is difficult to accurately predict whether a transmission will occur in the tail time. For example, in order to exploit the fast dormancy feature for user-interactive apps, it is necessary to predict when user will interact with a mobile network (e.g., when user will access a web page from a browsing app). Existing dormancy prediction techniques [4], [5], however, cannot predict when a next transmission will occur in the tail time for user-interactive apps. Our proposed technique can improve the prediction accuracy of tail-time transmissions significantly by considering personalized network usage characteristics for the user-interactive apps based on our fine-grained network activity classification.

3 Per-User Usage Characteristics in Smartphone Network Activities

In order to understand how smartphones consume radio energy while interacting with a mobile network, we performed a detailed quantitative analysis of the collected network usage logs from 25 smartphone users in Seoul. 25 study participants aged between 20 and 40, who used a 3G network as their main radio network, represented diverse user groups including college students, graduate students, bankers and kindergarten teachers. We distributed a modified Dalvik VM to the study participants and collected network usage logs over a period of 2 weeks. Since our user study was conducted without any detailed smartphone usage direction, they could use smartphones as usual.

For each user, we collected detailed network transmission logs and other useful information on network usage. For example, we collected important app execution events (such as a start and an end of each app), network traffic information (such as an app name and a packet size for each transmission), call stack snapshot for each socket API call, battery usage events (such as a charge/discharge state and a voltage level of each status change), and screen on/off events with their respective timestamps. For privacy reason, we did not collect detailed packet contents.

3.1 Radio Energy Consumption Breakdown in Smartphones

In order to understand how much radio energy is wasted in the tail time, we analyzed inter-transmission intervals from collected transmission logs. Since the main role of the tail time is to service closely clustered data transmissions quickly while consuming standby energy for distant future transmissions, we broke down the radio energy consumption as three parts, *data transmission energy*, *standby energy*, and *wasted tail energy*. Fig. 3 illustrates how we compute three energy components in the tail time. The data transmission energy is defined as the radio energy consumed in sending/receiving data within the tail time (e.g., τ_1 and τ_2 transmissions in Fig. 3). When a next transmission occurs

1. We present our measurement results for a 4G LTE device in Appendix A (available in our on-line supplemental material, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TMC.2015.2431712).

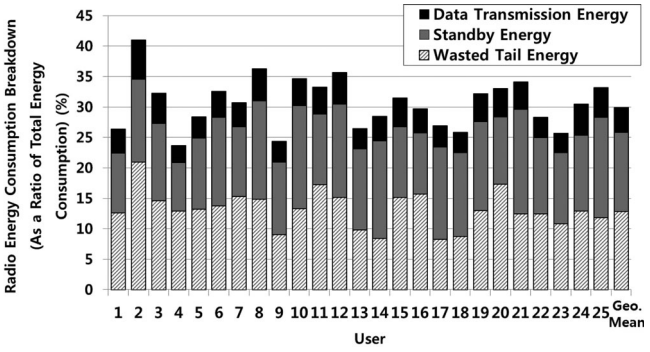


Fig. 4. Breakdown of radio energy consumption on smartphones.

within the tail time (e.g., τ_3 within T_1 in Fig. 3), we classify the energy consumed waiting for the next transmission as the standby energy. When there is no following next transmission to the end of the tail time (e.g., T_2 in Fig. 3), the wasted energy in the tail time is classified as the wasted tail energy. Using our 3G power model, which was constructed from our smartphone power measurement study as explained in Section 2, we computed three radio energy components of the radio energy consumption from the collected transmission logs.

In order to understand the impact of radio energy on the total smartphone energy consumption, we also estimated the total smartphone energy consumption using collected battery state logs. Whenever there is a change in battery voltage, we collected a new voltage level, a time stamp and charge/discharge state information. By estimating battery discharge rates for each point of battery voltage change and calculating discharged energy from the total battery capacity [11], we compute the total energy consumption while the battery is discharged.[2] In our analysis, we estimate how much energy is responsible for the radio device by comparing the total energy consumption with the radio energy consumption for the same discharge periods.

The radio energy consumption analysis results are summarized in Fig. 4. The figure shows that, as a ratio of the *total* smartphone energy consumption, how much radio energy is consumed for each energy component in the tail time. On average, the radio device consumed about 30 percent of the total battery energy while about 13 percent of the total energy consumption was wasted in the tail time. Actual data transmissions accounted only 4 percent of the total energy consumption. In the case of user 2, about 41 percent of the total energy consumption was from the radio device. From Fig. 4, it is clear that reducing the wasted tail energy will significantly improve an overall system-level energy efficiency as well as a radio energy efficiency.

3.2 User-Specific Network Activity Characteristics

In smartphones, network transmissions are initiated from a particular network activity. Thus, it is a logical assumption that a network activity is closely related to a network transmission trend in the tail time. For example, when a music streaming app finishes downloading a song, another transmission is unlikely to occur in the tail time because a smartphone user is very likely to listen to the just downloaded song. If we can accurately predict whether a following transmission will occur in the tail time or not based on a tail-time transmission trend of a network activity, we can reduce a large amount of energy wasted in the tail time by effectively exploiting the fast dormancy feature. In this section, we validate our hypothesis on a strong personalized connection between network activities and their tail transmission characteristics using 25 user logs we have collected.

In order to understand the transmission trends in the tail time, we consider two different aspects of correlationship between network activities and their transmission characteristics in the tail time from smartphone users' perspective. First, different users may exhibit different tail transmission trends for the same network activity. Since a smartphone is a highly personalized device, different smartphone users exhibit distinct usage patterns among themselves [12], [15], [16]. We expect, therefore, that different users may interact with a radio network in a completely different fashion. For example, when two people exchange messages each other using the same messenger app, a transmission trend may be quite different even for the same network activity (e.g., sending an instant message). One user may send several messages very quickly and frequently, and as a result, many transmissions are likely to occur in the tail time. On the other hand, if the other user reacts slowly to received messages and sends a very small number of messages over a long period of time, a transmission is unlikely to occur in the tail time for the same network activity.

Second, the same user responds very similarly to the same network activity with a *distinct tail transmission trend*. Because the same network activity is related in the same task and a smartphone user behaves quite similarly for the same network activity, we can expect that, the same user will react to the same network activity in a similar fashion, resulting in a distinct per-user transmission trend in the tail time. For example, suppose user receives notification messages from an unpopular online game app. Whenever a notification message is received, if the user is not interested in the game app, it is unlikely for the user to react to the network activity of receiving a notification message in the tail time.

Intuitively, our hypothesis on a strong per-user correlation between a network activity and its tail transmission pattern was motivated by an observation that a network activity forms a semantically meaningful network action (e.g., sending an instant message) *from user's perspective*. In order to validate our hypothesis, we have analyzed network activities from each collected log so that we can investigate transmission trends of network activities. Since a network activity leads to a series of clustered transmissions for a short time interval, we group a series of clustered transmissions in the same app as a single network activity. In our analysis, if two consecutive transmissions occur in less than 1-second time difference, we group two transmissions as a single network activity. (We evaluated different time difference values from 1 second to 3 seconds, but there were only negligible changes in the analysis results.) Since there were many different network activities in the collected

2. In our in-house energy measurement study, we have observed that PowerTutor [11] can accurately estimate the total energy consumption of a long time period. For example, the error rate of the estimated total energy consumption for 1 hour is less than 9 percent compared to measured energy consumption. Thus, we have concluded that the total power consumption of the 2-week's trace is accurate estimated.

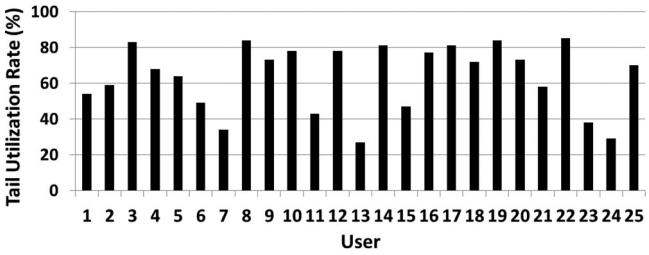


Fig. 5. TUR Variations over different users for the same network activity (sending an instant message).

transmission logs, we identify different network activities using their call stack information when each network activity was initiated. For example, if a transmission is initiated from the 'sendMessage()' function of a messenger app, we call the corresponding network activity (which initiated the transmission) as the sending an instant message network activity.[3]

In order to represent a tail-time transmission characteristic in a quantitative fashion, we define the tail utilization rate (TUR) for each network activity. For a given network activity $\mathbb{A}$ in a collected transmission log $\mathcal{L}_i$ for user i, $\mathrm{TUR}(\mathbb{A}|\mathcal{L}_i)$ is computed as follows:

$$\mathrm{TUR}(\mathbb{A}|\mathcal{L}_i) = \frac{N^{\mathbb{A}|\mathcal{L}_i}_{tail_{utilized}}}{N^{\mathbb{A}|\mathcal{L}_i}_{total}},$$

where $N^{\mathbb{A}|\mathcal{L}_i}_{total}$ is the total number of occurrences of the network activity $\mathbb{A}$ in the collected transmission log $\mathcal{L}_i$, and $N^{\mathbb{A}|\mathcal{L}_i}_{tail_{utilized}}$ is the total number of times that a following next transmission occurred in the tail time of $\mathbb{A}$. The higher $\mathrm{TUR}(\mathbb{A}|\mathcal{L}_i)$ is, the better utilized the tail time for the $\mathbb{A}$ (i.e., the more effectively utilizing the tail time).

3.2.1 Tail Transmission Variability over Different Users

In our analysis from the collected users' logs, we have found that there is a strong personalized tendency on network transmissions in tail times for the same network activity. We analyzed 25 users' TURs for the same network activity, sending an instant message, from a messenger app which is one of the most famous apps in Korea. As shown in Fig. 5, we observed that $\mathrm{TUR}(\mathbb{A}|\mathcal{L}_i)$ values for users $1 \leq i \leq 25$ for the same network activity $\mathbb{A}$ are significantly different among different users. For instance, for user 22 (who showed the highest TUR value), we can guess user 22 tends to check his/her messages frequently and react to them quickly. On the other hand, for user 13 (who had the lowest TUR value), we can assume that user 13 reacts very slowly to messages. Clearly, for user 13, a large amount of energy is wasted in the tail time. In order to take into account of these strong personalized network-usage characteristics, our proposed technique employs a user-specific online prediction model for transmission trends.

3. In order to identify network activities, we analyzed source codes of several open-source apps such as Google browser app. For example, we found that 'onPageLoad()' function initiates network traffics for browsing a web page. For non-open source apps, we also investigated which functions were called for its network activities while the apps were executed on our experimental smartphones.

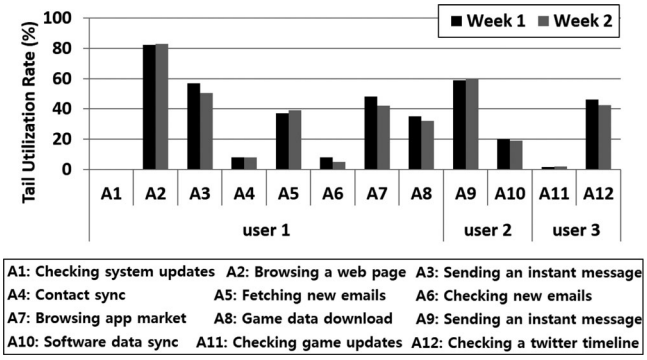


Fig. 6. Weekly per-user tail utilization rates comparison for twelve representative network activities.

3.2.2 Per-User Tail Transmission Characteristics of Different Network Activities

In order to verify whether the same user reacts to the same network activity with a similar tail-time transmission trend, we partitioned the collected log $\mathcal{L}_i$ into n sub-logs, $\mathcal{L}_{i_1}, \ldots, \mathcal{L}_{i_n}$ based on the log collection time. For example, in order to compare week-by-week characteristics of $\mathcal{L}_i$, we created two sub-logs, $\mathcal{L}_{i_1}$ and $\mathcal{L}_{i_2}$. Since the collected logs contained 2-week's network traces, $\mathcal{L}_{i_1}$ and $\mathcal{L}_{i_2}$ contains the first week's log and second week's log, respectively. For day-by-day comparisons, we partitioned $\mathcal{L}_i$ into 14 sub-logs in a similar fashion.

The result, summarized in Fig. 6, shows that the per-user TUR difference between two weekly TURs is very small for 12 representative network activities. Because of a space limit, we show the results of only three representative users, but other users' results are very similar to three users. The small TUR fluctuations can be observed even for user-interactive network activities such as $\mathrm{A3}(=$ sending an instant message$)$ as well as system-related network activities such as $\mathrm{A1}(=$ checking system updates$)$. As shown in Fig. 7, for the day-by-day tail utilization rates comparison, we have also founded per-user TUR fluctuations are quite small. Small TUR fluctuations for the same network activities strongly suggest that there is a unique per-user transmission trend for each network activity.

In addition, although the per-user TUR difference over different time periods is quite small for a given network activity specific, TUR values for different network activities are significantly different. For example, in Fig. 6, TUR of $\mathrm{A1}(=$ checking system updates$)$ is almost zero (i.e., almost no transmission occurs in the tail time.), thus the tail time is

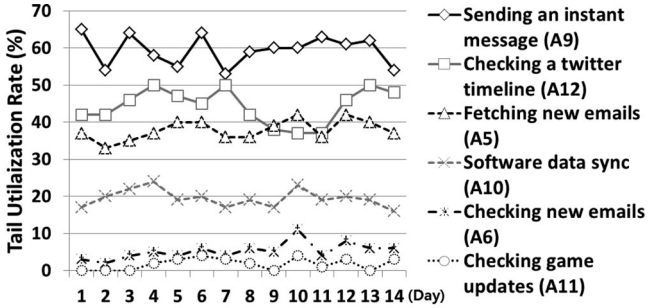


Fig. 7. Day-by-day per-user tail utilization rates comparison for six representative network activities.

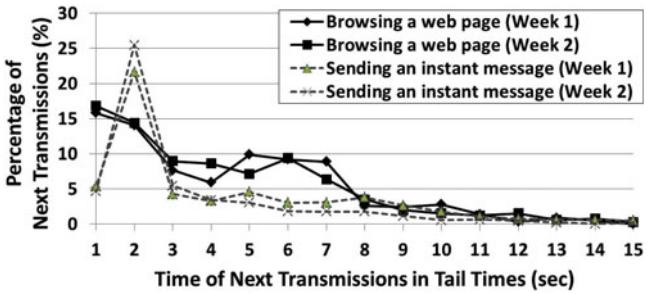


Fig. 8. Distributions of time of next transmissions in tail times for two network activities.

unnecessary for such network activities. On the other hand, when the network activity of $A2(=$ browsing a web page$)$ is completed, it is very likely that a next transmission occurs in the tail time. From this analysis result, we observed that it is important to predict the tail transmission taking account of per-user varying network transmission behavior over different network activities.

We have also observed that even when TUR for a given network activity is high, actual distributions of transmissions in the tail time are quite skewed. Fig. 8 illustrates this observation using two network activities, browsing a web page and sending an instant message. Although two network activities have high TUR values as shown in Fig. 6, the distributions of time of next transmission in tail times for the network activities are skewed to the right. For example, after browsing a web page is completed, most next transmissions happened within the first 8 seconds of the tail time. Fig. 8 also shows that the skewness in time of next transmissions in the tail time is preserved over week-by-week comparisons.

This result presents that user's behavioral tendency is reflected to the transmission trend of a network activity. Even though a network activity is very interactive (like sending an instant message), we can predict whether or when a next transmission will occur in the tail time based on the personalized tail-time transmission pattern. Our proposed technique exploits these skewed distributions in determining the likelihood of a transmission occurrence, for example, after x seconds in the tail time.

4 Design and Implementation of Personalized Diapause

4.1 Overview of Personalized Diapause

In this section, we describe our proposed smartphone radio energy optimization technique, pD, in detail. Exploiting user-specific tail-time transmission characteristics of each network activity as discussed in Section 3.2, our proposed technique makes intelligent on-line decisions on whether or when a transmission will occur in the tail time. The pD technique chooses candidate subintervals of the tail-time interval where a radio connection overhead is less than a given threshold value, and invokes the fast dormancy feature at a time point among the candidate subintervals where an energy saving is maximized. To this end, the pD technique systematically extracts meaningful network activities from call stack information of running apps, and learns tail transmission patterns for each extracted network activity.

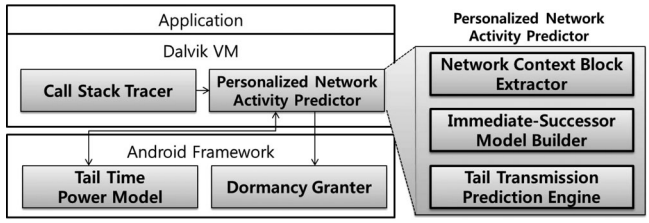


Fig. 9. An architectural overview of personalized diapause.

Fig. 9 shows an architectural overview of the pD technique. The personalized predictor for network activity module, which was added as an additional module to the Dalvik VM, is responsible for implementing the pD technique. Whenever the call stack tracer identifies a network transmission, the call stack information of the network transmission is sent to the network context block (NCB) extractor module where closely clustered network transmissions are grouped into a network activity. Then, the immediate-successor model builder module constructs an immediate-successor model which represents a tail transmission pattern for each network activity. An immediate-successor model of a network activity represents a tail transmission pattern. Using an immediate-successor model, the tail transmission prediction engine module determines when to invoke the fast dormancy feature based on the tail time power model. Finally, the dormancy granter module invokes the fast dormancy feature when requested from the personalized network activity predictor module.

4.2 Network Context Block Extractor

In order to learn a tail transmission pattern automatically for each network activity, our pD technique systematically extracts meaningful network activities from call stacks of running apps. The network context block extractor module groups a series of inter-related network transmissions into *network context block* which represents a meaningful network activity unit. A key observation is that same network activities are initiated from same execution paths represented by same function call stacks. Based on the observation, we keep track of call stacks for a series of transmissions, and identify its network activity by exploiting the monitored call stack information. We explain how we distinguish different network activities in Section 4.2.1, and we validate our network activity extraction technique in Section 4.2.2.

4.2.1 Automatic Extraction of Network Activity

In order to extract meaningful network activities, the network context block extractor module keeps track of network transmissions initiated from socket APIs. We partition a series of the network transmissions into a sequence of *network context blocks*. Two adjacent network context blocks are distinguished by an idle transmission interval whose length is longer than a predefined threshold value. Thus, the main procedure of the network context block extractor module is to classify a series of network transmissions into network context blocks which will be used as a basic unit in predicting future transmissions.

The network context block extractor module distinguishes different network activities by identifying program execution paths which initiate each transmissions in

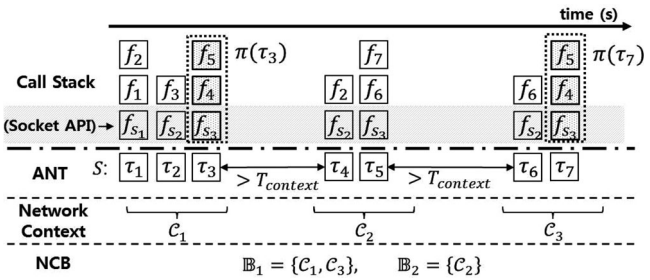


Fig. 10. An example of extracting network context blocks.

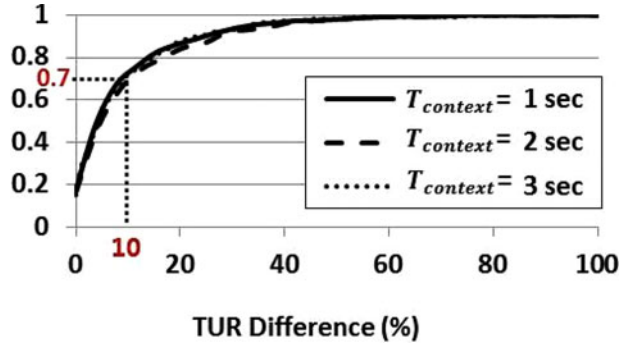


Fig. 11. CDF of TUR differences for [user, NCB] combinations over 2 Weeks (total number of [user, NCB]'s: 2,681, 2,155, 2,011 for $T_{context} = 1, 2, 3$ sec., respectively).

network activities [6]. Since a network transmission request of an app is eventually sent to one of socket API functions, we distinguish different network activities at the socket API level. We define an *atomic network transmission* (ANT) as a network data transfer initiated from a socket API function. For example, the socket API functions such as connect, write, read, send and recv can initiate different ANTs. In order to distinguish different ANTs, once an ANT is initiated from a socket API, we associate each ANT with its unique ID, called as ANT-ID. The ANT-ID $\pi(\tau_i)$ of an ANT τ_i is computed by summing the addresses of functions in the call stack that lead to the socket API function that initiates the corresponding ANT τ_i. Thus, an ANT-ID identifies an execution path where an ANT is initiated. The call stack tracer sends an ANT-ID to the network context block extractor module when a socket API initiates an ANT.

The network context block extractor module views network transmissions of an app A as a sequence $\mathcal{S}_A$ of ANTs (sent by the call stack tracer), i.e., $\mathcal{S}_A = \langle \tau_1, \ldots, \tau_n \rangle$ where τ_i is an ANT and τ_i happens before τ_j if $i < j$. In order to identify a series of consecutive ANTs which are involved in a single network activity, we partition the sequence of ANTs into a sequence of network contexts. Given a sequence $\mathcal{S}_A$ of ANTs, we construct a sequence $\mathcal{C}_A$ of network contexts, $\mathcal{C}_A = \langle \mathcal{C}_1^A, \ldots, \mathcal{C}_k^A \rangle$, where each $\mathcal{C}_i^A$ is a network context. A network context $\mathcal{C}_i^A$ consists of successive ANTs $\langle \tau_{i_1}, \ldots, \tau_{i_k} \rangle$ where the inter-ANT interval between two consecutive ANTs is less than a threshold time $T_{context}$. That is, there exists an idle transmission whose length is at least $T_{context}$ between any two $\mathcal{C}_i^A$ and $\mathcal{C}_j^A$. Thus, we can distinguish each network context $\mathcal{C}_i^A$ at runtime whenever an idle transmission interval of $T_{context}$ length is observed. Intuitively, each network context represents a meaningful network activity such as an activity of downloading a song.

Since we represent a network context by a sequence of ANTs, even if two network activities $\mathcal{C}_i$ and $\mathcal{C}_j$ consist of *almost* identical ANT-IDs except for one or two different ANTs, $\mathcal{C}_i$ and $\mathcal{C}_j$ considered as different network activities. For example, consider two network activities $\mathcal{C}_1$ and $\mathcal{C}_2$ which are both downloading a song network activities. If $\mathcal{C}_1$ downloads a song with a text data of lyrics while $\mathcal{C}_2$ downloads a song without a text data of lyrics, a sequence of ANT-IDs for $\mathcal{C}_1$ is different from that for $\mathcal{C}_2$ because $\mathcal{C}_1$ needs to download a text data. However, $\mathcal{C}_1$ and $\mathcal{C}_2$ are semantically almost equivalent network activities, resulting in almost identical tail transmission patterns. In order to make $\mathcal{C}_1$ and $\mathcal{C}_2$ to be equivalent network activities, we define two network contexts, $\mathcal{C}_i^A$ and $\mathcal{C}_j^A$, are equivalent if at least one ANT-ID $\pi(\tau_{i_p})$ of τ_{i_p} in $\mathcal{C}_i^A$ is equivalent to ANT-ID $\pi(\tau_{j_q})$ of τ_{j_q} in $\mathcal{C}_j^A$. A group of equivalent network contexts is called an *(equivalent) network context block*. Thus, each network context in the same NCB indicates a semantically similar network activity.

Fig. 10 illustrates how the network context block extractor module automatically extracts NCB using an example. Given a sequence of ANTs $\mathcal{S} = \langle \tau_1, \ldots, \tau_7 \rangle$, whenever there is an idle transmission interval of $T_{context}$ length, we partition $\mathcal{S}$ into three network contexts, $\mathcal{C}_1, \mathcal{C}_2$ and $\mathcal{C}_3$. Since τ_3 and τ_7 have the same execution path, τ_3 and τ_7 have the same ANT-ID, i.e., $\pi(\tau_3) = \pi(\tau_7)$. Therefore, $\mathcal{C}_1$ and $\mathcal{C}_3$ belong to the same NCB $\mathbb{B}_1 = \{\mathcal{C}_1, \mathcal{C}_3\}$. On the other hand, $\mathcal{C}_2$ forms its own NCB $\mathbb{B}_2 = \{\mathcal{C}_2\}$. Two equivalent network contexts, $\mathcal{C}_1$ and $\mathcal{C}_3$ in $\mathbb{B}_1$, are assumed to perform the same network activity. For example, if $\mathcal{C}_1$ were used for the network activity, sending an instant message, $\mathcal{C}_3$ would be assumed to do the same network activity, sending an instant message.

4.2.2 Validation of Network Context Block

In order to validate if our proposed NCBs are good monitoring units in representing tail transmission characteristics for user's network activities, we have extracted all NCBs from the collected user logs using three different $T_{context}$ values, 1, 2 and 3 seconds. Then, we have computed the tail utilization rates for every [user, NCB] combination and compared two weekly TURs for the same [user, NCB] combination. The results summarized in Fig. 11 show that, for most of [user, NCB] combinations, the TUR difference between two weekly TURs is very small. For example, about 70 percent of 2,681 [user, NCB] combinations extracted with $T_{context} = 1$ second differed in their weekly TURs by less than 10 percent. Small TUR fluctuations for the same [user, NCB] combinations strongly suggest that user tends to react in a similar fashion to the same network activity. Although more specific details on a particular network activity may be obtained by using user-level information (e.g., web addresses for browsing a web page), the results show that the proposed network context, which can be constructed only using system-level information, is appropriate in capturing tail transmission patterns for semantically meaningful network activities. As shown in Fig. 11, there are virtually no differences in TUR distributions among extracted [user,

NCB] combinations when different $T_{context}$ values were used. In the rest of the paper, we assume that $T_{context}$ is set to 1.

4.3 Immediate-Successor Model Builder

In order to predict whether a next transmission will occur or not in the tail time after a given network context in an NCB is completed, the immediate-successor model builder module builds an immediate-successor model for each NCB. The key step of the immediate-successor model builder module, therefore, is to construct a probability mass function of next transmission times in the tail time for each NCB (similar to one shown in Fig. 8). For each NCB invocation, we keep track of when a successive transmission occurs, and compute the inter-transmission interval between the network context and the successive transmission. By accumulating the intervals, the immediate-successor model for each NCB presents when and how frequently next transmissions occur in the tail time.

Before we describe the immediate-successor model builder module, we first define several terms which are useful in explaining the process of building a tail transmission model. For a network context $\mathcal{C}_i$, we call $\mathcal{C}_j$ the *immediate successor context* of $\mathcal{C}_i$ if $\mathcal{C}_j$ happens after $\mathcal{C}_i$ and there is no other network context between $\mathcal{C}_i$ and $\mathcal{C}_j$. In particular, we define the first ANT of the immediate successor context of a given network context $\mathcal{C}$ as the *immediate successor transmission* of a network context $\mathcal{C}$. For example in Fig. 10, the immediate successor context of the network context $\mathcal{C}_1$ is $\mathcal{C}_2$, and the immediate successor transmission of the network context $\mathcal{C}_1$ is τ_4. Similarly, given an NCB $\mathbb{B} = \{\mathcal{C}_1, \ldots, \mathcal{C}_l\}$, we define the immediate successor transmissions of $\mathbb{B}$ as a set of the immediate successor transmission of each $\mathcal{C}_i \in \mathbb{B}$. In the example shown in Fig. 10, the immediate successor transmission of $\mathbb{B}_2$ is $\{\tau_6\}$. In the rest of this paper, for 'immediate successor transmission' of a network context or an NCB, we use *IS* as a shorthand form.

Given an NCB $\mathbb{B}$, its immediate-successor model maintains when and how many times *IS*'s occur for all $\mathbb{B}$'s invocations. An immediate-successor model consists of $(N+1)$ *subintervals*, N equivalent-length subintervals $s_0, \ldots, s_{N-1}$ (for when *IS*'s occur in the tail time) and the other subinterval s_N (for when *IS*'s occur after the tail time). Each subinterval, $s_i = [b_i, e_i)$, is defined as follows:

$$b_i = i \cdot \frac{T_{tail}}{N} \quad i = 0, \ldots, N,$$

$$e_i = \begin{cases} (i+1) \cdot \frac{T_{tail}}{N}, & (i = 0, 1, \ldots, N-1), \\ \infty, & (i = N). \end{cases}$$

Each subinterval s_i keeps track of the number n_i of *IS*'s within $[b_i, e_i)$. For example, if an *IS* is occurred after T_{tail}, we accumulate their occurrences in n_N.

Fig. 12 illustrates the process of building an immediate-successor model for NCB $\mathbb{B}_1$. In this example, we assume that the network contexts $\mathcal{C}_1$ and $\mathcal{C}_2$ belong to the same NCB $\mathbb{B}_1$ while $\mathcal{C}_3$ belongs to a different NCB $\mathbb{B}_2$. Because the inter-context interval θ_1 between $\mathcal{C}_1$ and $\mathcal{C}_2$ is 4.3 (i.e., the first ANT of $\mathcal{C}_2$ occurs in the subinterval s_4 of the tail time interval after $\mathcal{C}_1$ has been completed.), n_4 is incremented. For the inter-context interval θ_2, n_7 is incremented

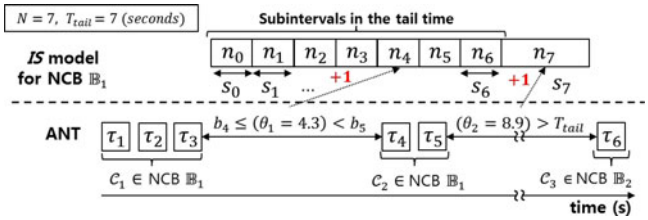


Fig. 12. An example of building an immediate-successor model.

because $\theta_2 > T_{tail}$. As expected, the immediate-successor model will approximate the probability mass function of *IS*'s for $\mathbb{B}_1$ when a large number of *IS*'s of $\mathbb{B}_1$ are accumulated.

4.4 Tail Transmission Prediction Engine

Once a network context in an NCB $\mathbb{B}$ is completed, based on an immediate-successor model for the NCB $\mathbb{B}$, the tail transmission prediction engine decides if it is *beneficial* to invoke the fast dormancy feature. First, the tail transmission prediction engine computes the *energy gain* for every possible scenario on future network usage, as described in Section 4.4.1. Second, the tail transmission prediction engine also computes *resource allocation overhead* in order to avoid excessive radio reconnection increase, as described in Section 4.4.2. Subsequently, the tail transmission prediction engine determines when to invoke the fast dormancy feature for maximizing the energy gain with the consideration of the resource allocation overhead.

In describing the detailed decision procedure of the tail transmission prediction engine, we use a following notation derived from an immediate-successor model. Given an immediate-successor model for an NCB $\mathbb{B}$, we define a probability value p_k which indicates the probability that an *IS* occurs in the subinterval s_k as follows:

$$p_k = \frac{n_k}{\sum_{j=0}^{N} n_j}.$$

4.4.1 Radio Energy Gain Analysis

The tail transmission prediction engine first computes the energy gain G_i, which represents the energy gain when a radio connection is released at b_i, for each $b_i \in \{b_0, \ldots, b_{N-1}\}$. Let $E_{subtail}$ represent the wasted radio energy within a single subinterval of the tail time. For example, in 3G network, $E_{subtail}$ can have two values according to its power state, $E^{DCH}_{subtail}$ for subintervals belonging to `DCH` state and $E^{FACH}_{subtail}$ for subintervals belonging to `FACH` state. If our proposed technique is applied for different networks (e.g., 4G LTE), we can easily support its radio power by changing the definition of $E_{subtail}$. In addition, E_{ohd} represents an additional energy overhead for re-establishing a radio connection. Using two constants, $E_{subtail}$ and E_{ohd}, Algorithm 1 describes how the energy gain G_i is computed for a given immediate-successor model of an NCB.

Fig. 13 illustrates how Algorithm 1 computes the energy gain G_4 as an example. When the fast dormancy feature is invoked at b_i, we consider $(N - i + 1)$ different cases for potential immediate successor transmission times: each $b_j \in \{b_i, \ldots, b_N\}$ (`Line 4`). Note that for any *IS* τ_j initiated at $[b_i, e_i)$, we assume that τ_j was initiated at b_i. This

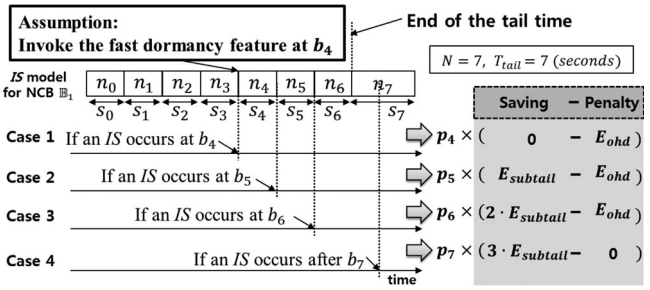


Fig. 13. An example of computing energy gain by tail transmission prediction engine.

simplifies procedures for computing energy gains and resource allocation overheads although it conservatively estimates an actual energy gain. In the example shown in Fig. 13, there are four possible cases when an *IS* occurs, namely at b_4, b_5, b_6, and b_7. For each case, we consider energy saving and penalty values (`Lines 6 and 7`). For our example in Fig. 13, there are four cases as follows:

- Case 1 (an *IS* occurs at b_4 immediately after a radio connection is disconnected at b_4): There is no energy saving, but the energy penalty E_{ohd} for re-establishing the radio connection should be paid.
- Case 2 (an *IS* occurs at b_5): It is possible to save the wasted tail energy between b_4 and e_4 which is equal to $E_{subtail}$. The energy penalty is E_{ohd}.
- Case 3 (an *IS* occurs at b_6): The wasted tail energy from b_4 to e_5, $2 \times E_{subtail}$, is saved, and the energy penalty is E_{ohd}.
- Case 4 (an *IS* occurs after the end of the tail time, b_7): We save entire wasted tail energy from b_4, $3 \times E_{subtail}$, without an energy penalty.

For each $b_j \in \{b_i, \ldots, b_N\}$, the probability that an *IS* occurs at b_j, is conservatively estimated by p_j. Thus, the expected energy gain for each case is computed by multiplying p_j and the difference of energy saving and energy penalty. We sum up the expected energy gain for all cases into β (`Line 8`). In order for the fast dormancy mode to be invoked at b_i, there should be no *IS* at $[b_0, e_{i-1})$, we compute the probability $(1 - q_k)$ that no *IS* occurs within $[b_0, e_{i-1})$ (`Line 10`). Lastly, the energy gain G_i is given by $(1 - q_i) \times \beta$ (`Line 11`).

Algorithm 1. Compute the Energy Gain G_i (Given a Fast-Dormancy-Invocation Time b_i)

```
1  // assumption: the fast dormancy
2  // feature is invoked at b_i
3  β ← 0;
4  for each b_j in {b_i, ..., b_N} do
5     // if an IS occurs at b_j
6     saving ← (j − i) × E_subtail;
7     penalty ← (j ≠ N) ? E_ohd : 0;
8     β ← β + p_j × (saving − penalty);
9  end
10 q_i ← (i ≠ 0) ? Σ_{k=0}^{i−1} p_k : 0;
11 G_i ← (1 − q_i) × β;
```

4.4.2 Radio Resource Allocation Overhead Analysis

From the viewpoint of energy saving in smartphones, it is more beneficial for `pD` to invoke the fast dormancy feature whenever the energy gain G_i is maximized. However, too frequent switches to the dormancy mode can incur high radio resource allocation overhead from frequent radio reconnections. As discussed in Section 2, an additional radio reconnection leads to a long delay for smartphone user, for example, about 2-second delay in the 3G network, and extra signaling overhead for mobile network. Thus, we should consider limiting excessive reconnection increases while exploiting the fast dormancy feature in saving wasted radio energy. In order to avoid the excessive radio resource allocation overhead, the tail transmission prediction module takes into account an upperbound on the increase in the number of radio reconnections in deciding to invoke the fast dormancy feature. We manage the frequency of dormancy mode switches so that the increase in the number of radio reconnections does not exceed $(1 + \delta)$ times over when no fast dormancy feature is used (where $0 < \delta$).

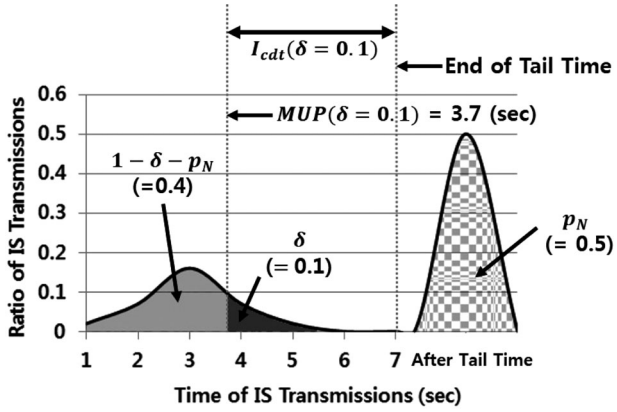


Fig. 14. An example of estimating radio reconnection increase by tail transmission prediction engine.

In order to limit the increase in the number of radio reconnections by less than $(1 + \delta)$ times over when no fast dormancy feature is used, we first compute a candidate interval $I_{cdt}(\delta)$, which we expect that the radio reconnection will not increase more than δ if the fast dormancy feature is invoked in the candidate interval. More formally, we select a candidate interval $I_{cdt}(\delta)$ so that the probability of an *IS* occurrence in the interval is less than δ. Fig. 14 describes the procedure of estimating radio reconnection increase and a candidate interval using an example. Given a probability model for immediate-successor occurrences, we define a modified upper $\delta \times 100\%$-percentile point, in short $MUP(\delta)$, as a time t where the cumulative probability from t to the end of the tail time matches δ. In the example of Fig. 14, $MUP(\delta = 0.1)$ is 3.7 seconds. That is, if we invoke the fast dormancy feature at 3.7 seconds, we expect that the probability of an *IS* occurrence in the remaining tail time is $\delta (= 0.1)$. Using $MUP(\delta)$, we compute the *candidate interval* $I_{cdt}(\delta)$ as $[MUP(\delta), b_N)$. Intuitively, if we invoke the fast dormancy within $I_{cdt}(\delta)$, we expect that the number of radio reconnections does not exceed more than $(\delta \times 100)\%$ over the no-fast-dormancy case.

Using the candidate interval and the computed energy gain, we finally determine the time in the candidate interval to invoke the fast dormancy feature for maximizing the energy gain. Algorithm 2 describes an overall procedure of determining when to invoke the fast dormancy feature for given δ and G_i for $0 \leq i < N$. `Lines 1~8` show how we compute $I_{cdt}(\delta)$ from the immediate successor model. Since

the immediate-successor model builder constructs a *discrete* probability model for *IS* occurrences, we conservatively estimate the candidate interval $I_{cdt}(\delta) = [b_x, b_N)$ where b_x is the smallest value such that $b_x \geq MUP(\delta)$. For example, in Fig. 14, the candidate interval $I_{cdt}(\delta = 0.1)$ is estimated by [4, 7). Then, as shown in `Lines 9∼14`, we choose b_m as the time to invoke the fast dormancy feature where G_m is the maximum energy gain in the $I_{cdt}(\delta)$. For example, if $I_{cdt}(\delta = 0.1)$ is estimated by $[4, 7)$, we consider three candidate energy gains, G_4, G_5, and G_6. If the maximum energy gain were G_5 among them, we would select b_5 as the time to invoke the fast dormancy feature.

Algorithm 2. Select the Time b_m to Invoke the Fast Dormancy Feature (Given δ and Energy Gain G_i)

```
1  // Step 1. compute I_cdt(δ).
2  a ← (N − 1);
3  While a ≥ 0 and δ ≥ Σ_{k=a}^{N−1} p_k do
4     // i.e., b_a ≥ MUP(δ)
5     a ← a − 1;
6  end
7  b_x ← b_{a+1};
8  I_cdt(δ) ← [b_x, b_N);
9  // Step 2. choose b_m in I_cdt(δ) = [b_x, b_N)
10 // for maximizing energy gain.
11 m ← argmax_{i∈[b_x,b_N)} G_i;
12 if G_m > 0 then
13    Select b_m;
14 end
```

Since an immediate successor model is a probability model, actual *IS* occurrences may not follow the immediate successor model of an NCB. Therefore, in the worst case, the increase in the number of radio reconnections may exceed more than $(1 + \delta)$ times over no-fast-dormancy case. In order to guarantee that no such case can happen, pD computes the number of radio reconnections under no-fast-dormancy case as well. Comparing the number of reconnections in pD with that of no-fast-dormancy case, we do not invoke the fast dormancy feature if pD increases radio reconnections more than $(1 + \delta)$ times over no-fast-dormancy case.

Handling concurrent transmissions from multiple apps. Since multiple apps can share the current radio connection at the same time, before invoking the fast dormancy feature, we also make sure that it will be safe for these apps to disconnect the radio connection. In our implementation, which is similar to one proposed in [4], the dormancy granter module (described in Fig. 9) first checks whether all recent apps (which used radio transmissions) request the dormancy mode changes. When all of them want to switch to the dormancy mode, the fast dormancy feature is actually invoked to the radio device. In the current implementation, an app is classified as a *recent* app if less than the tail time has elapsed since the app's last transmission.

5 Experimental Results

5.1 Experimental Environment

In order to evaluate the efficiency of the proposed pD technique, we have implemented pD on Nexus S Android reference smartphones running Android 2.3 (Gingerbread). The pD modules, which are described in Section 4.1, were implemented in the Dalvik VM and Android framework, and they contain about 2,000 lines of C and Java code. The current implementation of pD was carefully optimized so that user does not notice any change in the execution time of an app when pD is supported. For example, a function of tracking socket APIs and call stack information, which is one of performance-critical functions of pD, was implemented by adding only 50 new/modified lines to the original Dalvik VM code.

TABLE 1
Parameters Used for Experimental Evaluations

Name	Description	Value
$T_{context}$	The maximum interval between two consecutive ANTs belonging to the same network context	1 sec
N	The number of subintervals in the tail time	15
T_{tail}	The length of the tail time	15 secs
$E^{DCH}_{subtail}$	The subinterval energy consumption in the DCH state	820 mJ
$E^{FACH}_{subtail}$	The subinterval energy consumption in the FACH state	410 mJ
E_{ohd}	The energy overhead for a radio reconnection	2650 mJ

Even though the implemented pD technique can make online decisions while user uses a smartphone, in order to evaluate different policies in our experiments, we have used an additional custom log replayer tool for reproducing the collected logs from 25 users. The custom log replayer tool reproduces the collected call stack logs to the personalizd network predictor module. Thus, the personalized network activity predictor module can determine when to invoke the fast dormancy feature in the same way as the online decisions.

Table 1 summarizes various parameters used in our experiments. For 3G radio energy parameters, T_{tail}, $E^{DCH}_{subtail}$, $E^{FACH}_{subtail}$ and E_{ohd}, we used actual measurement values from our test smartphones (i.e., Nexus S smartphones). $T_{context}$ was set to 1 second (as explained in Section 4.2.2). The tail time was divided into 15 subintervals (i.e., 1 second per a subinterval) which give a sufficient granularity to distinguish different tail-time transmission patterns among different users.

In order to evaluate the running overhead of the proposed pD technique, we conducted a quantitative analysis study for the execution time overheads. We measured the additional time overhead, which includes extracting NCBs, building immediate successor model and predicting tail transmissions, while the 25 collected logs were replayed on Nexus S Android reference smartphones. Fig. 16a presents the impact of pD on the total execution time of apps. The result shows that our pD technique only took 0.01 percent of the total execution time of apps on average. In addition, as shown in Fig. 16b, the average time overhead per network context was less than 0.25 ms. Since the Dalvik VM already keeps track of function call chains and call stack information, there is little execution time overhead in extracting NCBs. Since the average number of NCBs per app was

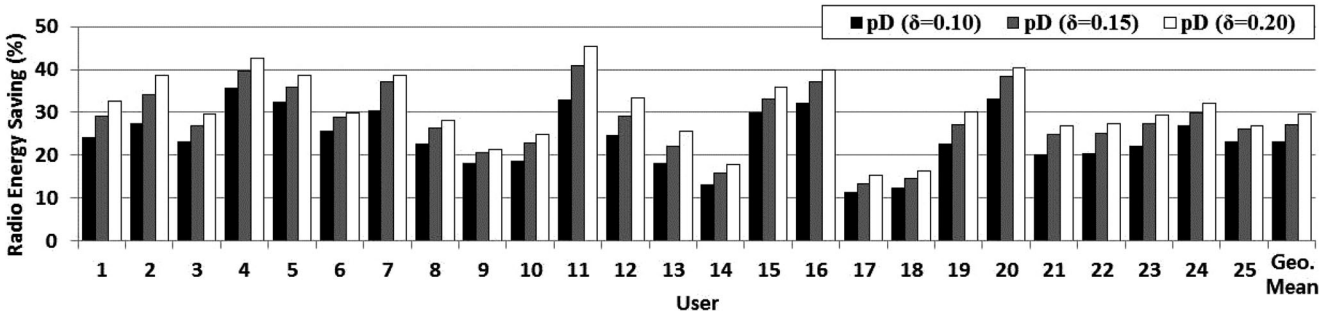


Fig. 15. Per-user radio energy saving comparisons under different δ's.

about 4, additional time overheads of building immediate successor models and predicting tail transmissions were also small. We also measured the actual energy consumption while running the pD technique, and observed that the extra energy overhead of pD is also less than 0.01 percent of the total energy consumption of apps on average.

5.2 Per-User Radio Energy Saving Comparisons

Fig. 15 shows the impact of the pD technique on the total radio energy consumption for all 25 users. Using no-fast-dormancy support as a baseline (which we denote by baseline), Fig. 15 shows how pD's radio energy saving ratios change over different δ's. The pD technique was evaluated under the assumption that the increase in the number of radio reconnections is limited by 10, 15 and 20 percent, respectively (i.e., $\delta = 0.10, 0.15$ and 0.20). The result shows that pD can save the radio energy on average by 23 percent over the no-fast-dormancy case with $\delta = 0.10$. For user 11, the maximum energy saving of 33 percent is achieved over the no-fast-dormancy case. For $\delta = 0.15$ and $\delta = 0.20$, since there are more opportunities for switching to the dormancy mode by invoking the fast dormancy more aggressively, pD saves more radio energy, on average, by 27 and 30 percent over the baseline technique, respectively.

We have also evaluated the impact of the proposed pD technique on the energy consumption of the whole system. Based on our analysis study for the total energy consumption described in Section 3.1, we computed how much the total energy consumption can be saved by exploiting the pD technique. In the evaluation, we observed that pD can reduce the energy consumption of the whole system on average by about 7 percent with less than 10 percent reconnection increase. For $\delta = 0.15$ and 0.20, the total smartphone energy consumption is saved on average by 8 and 9 percent, respectively.

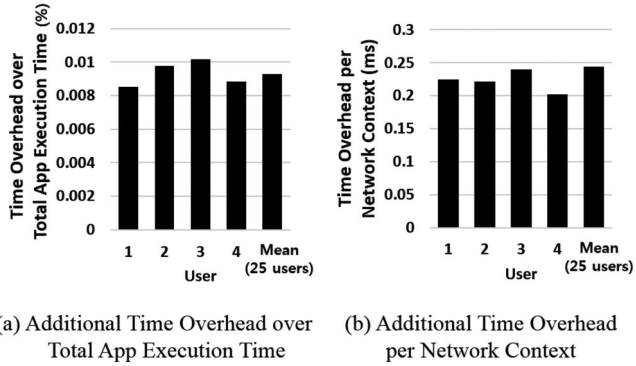
5.3 Radio Energy Saving Breakdown

In order to better understand the effect of the pD technique on radio energy saving, we further classified the radio energy consumption as three components (as described in Section 3.1). Fig. 17 shows radio energy saving breakdowns for four representative users. Because of a space limit, we present the radio energy saving breakdowns for four users. The result shows that pD reduces the radio energy consumption by saving the wasted tail energy. For $\delta = 0.10$, 0.15 and 0.20, the wasted tail energy was reduced by 51, 59, and 63 percent, respectively, over the baseline technique.

On the other hand, the standby energy was saved, on average, by only about 5, 6, and 8 percent, respectively, for $\delta = 0.10$, 0.15 and 0.20 over baseline. Since pD is effective when there is no *IS*'s in the tail time, it is difficult to reduce standby energy which is consumed when there is an *IS* in the tail time. A small reduction in the standby energy consumption, however, is possible because pD may decide to release a radio connection even when an *IS* occurs in the tail time. As shown in Fig. 17, pD is more likely to take such chances with a higher δ, thus saving more standby energy.

5.4 Energy versus Reconnection Increase Tradeoff

In order to understand how pD manages the radio reconnection increase for different δ's, we calculated the increase in the number of reconnections over baseline. As shown in Fig. 18, pD fully utilizes a given reconnection budget δ in order to achieve higher energy saving. For example, for four users, pD increases the number of reconnections by



(a) Additional Time Overhead over Total App Execution Time (b) Additional Time Overhead per Network Context

Fig. 16. Running overhead of pD technique.

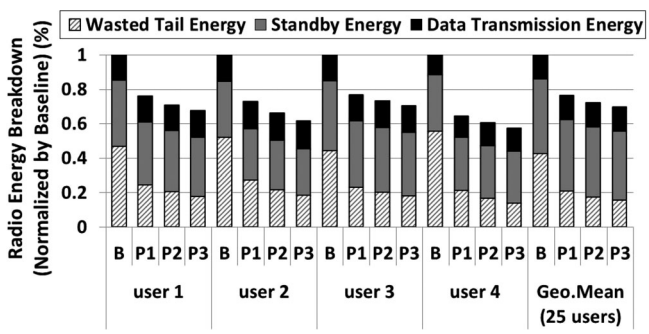


Fig. 17. Breakdown of radio energy saving (B: baseline, P1: pD($\delta = 0.10$), P2: pD($\delta = 0.15$), and P3: pD($\delta = 0.20$)).

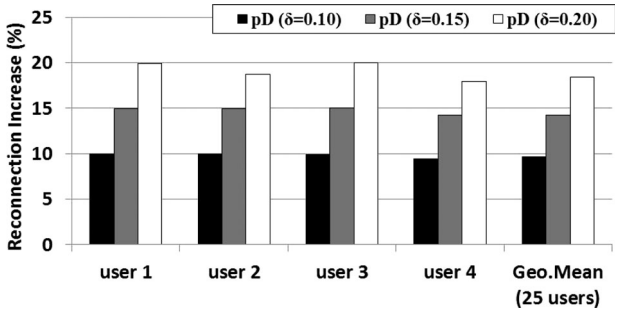


Fig. 18. Changes in the number of radio network reconnections under different δ's.

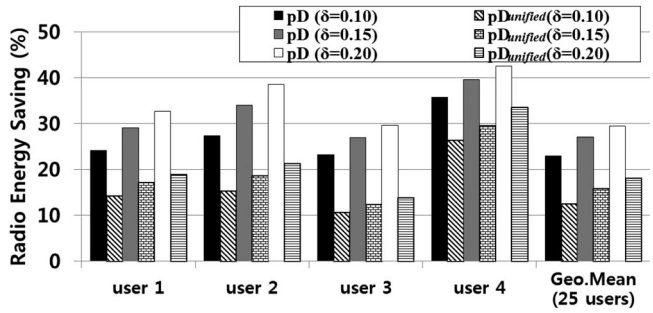


Fig. 20. Effect of NCB extraction on radio energy saving.

9.99, 9.97, 9.91 percent and 9.43 percent when δ was set to 0.10. Overall, for different δ's, 0.10, 0.15 and 0.20, respectively, the number of reconnections increases, on average, by 9.6, 14.2 and 18.4 percent, respectively. These results clearly show that pD makes sufficient use of a given δ by effectively adjusting to changing different transmission patterns over different users.

Note that the proposed pD, which improves the energy efficiency of mobile devices, can be beneficial to mobile network service providers as well. For example, since pD allows mobile devices to be disconnected from a mobile network under a controlled reconnection increase, extra network reconnections can be compensated by decreased connection times by mobile devices.

5.5 Energy Saving for Different Network Activities

We have also analyzed the impact of the pD technique on energy consumption for different network activities. Fig. 19 shows a breakdown of the radio energy consumption for five representative NCBs extracted from user 1's log when $\delta = 0.10$. In order to show the tail-time transmission trend of each NCB, the TUR value of each NCB is included together. The result shows that pD adaptively saves radio energy by considering varying tail transmission trends over different network activities. Since a low TUR value represents that a large portion of radio energy is wasted in the tail time, there is more opportunity for reducing energy consumption. For example, for NCB $\mathbb{B}_1$ whose TUR value is 11 percent, the wasted tail energy of baseline is responsible for about 79 percent of the total radio energy consumption. Thus, by taking account of the tail transmission trend of NCB $\mathbb{B}_1$, pD reduces the wasted tail energy by 74 percent over the baseline technique. On the other hand, because the radio energy is rarely wasted for NCB $\mathbb{B}_5$, pD saves only 26 percent of the wasted tail energy.

Furthermore, the result in Fig. 19 shows that some portion of the wasted tail energy is not saved even by pD. For example, even though the TUR of NCB $\mathbb{B}_1$ is 11 percent, about 20 percent of the tail energy is still wasted under pD. This inefficiency of pD comes mainly from when the pD technique decides a dormancy mode switch *in the latter part* of the tail time when $MUP(\delta)$ is close to the end of the tail time interval.

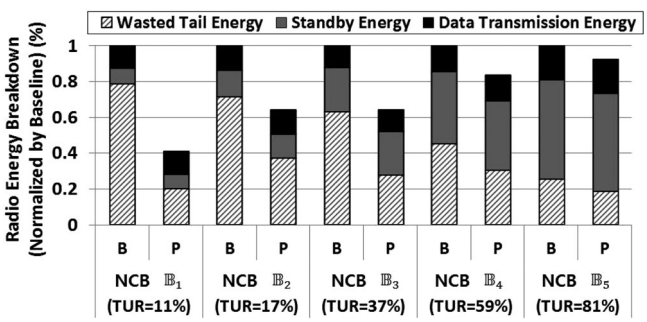


Fig. 19. Radio energy saving breakdown of five representative NCBs for user 1 (B: baseline and P: pD($\delta = 0.10$)).

5.6 Fine-Grained Network Activity Extraction

In order to understand the impact of our proposed fine-grained NCB classification technique on radio energy savings, we compared the pD technique with $\text{pD}_{unified}$, a simplified version of pD.[4] The $\text{pD}_{unified}$ technique assumes that all NCBs from a single user share a single immediate-successor model. That is, $\text{pD}_{unified}$ does assume that user interacts with different network activities in a similar fashion. Fig. 20 shows that pD saves more energy over $\text{pD}_{unified}$. For example, pD achieves on average 84 percent higher energy saving over $\text{pD}_{unified}$ with $\delta = 0.10$. This comparison shows that a fine-grained NCB separation based on semantic differences even for a single user is important in achieving a high energy efficiency.

5.7 Combining pD with Other Context Information

The pD technique can be further improved in various fashions by exploiting other useful context information. For example, if the pD technique knows *a priori* that user does not experience any inconvenience even though there are a large number of radio reconnections, the pD technique may decide to use a large δ's in favor of a higher energy saving. In this section, we present a simple case study of such an extension to the proposed pD technique using the display status as an additional context hint.

When the display screen of a smartphone is turned off, extra delays do not affect the user's experience [14]. Therefore, it makes sense to invoke the fast dormancy feature more aggressively, thus improving the overall radio energy efficiency without a negative effect on the user experience.

4. In Appendix B, available in the online supplemental material, we compared pD with two other simple dormancy techniques. For a better explanation that pD saves radio energy based on its accurate prediction, we present comparison results with a timeout-based approach. In addition, in order to demonstrate the impact of our online model building, we compare our technique with a simplified pD which predicts future transmissions based on a static model without any online learning.

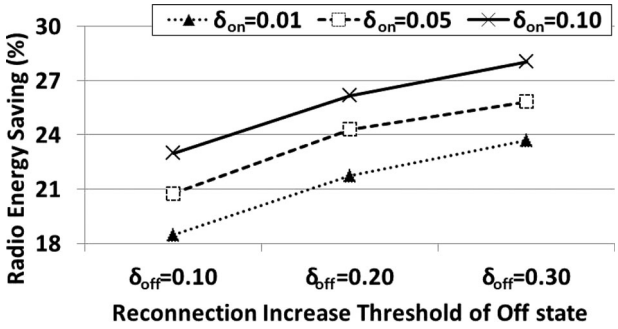


Fig. 21. Effect on energy saving of $\texttt{pD}_{UXAware}$ for different $(\delta_{on}, \delta_{off})$'s.

For example, in our collected user logs, about 60 percent of the total radio energy was consumed during the screen-off period. Thus, if we reduce radio energy more aggressively during screen-off periods over screen-on periods, we reduce the radio energy consumption significantly more without affecting the user experience.

Based on this idea, we have implemented a variant of the pD technique, called $\texttt{pD}_{UXAware}$. In determining whether or when to invoke the fast dormancy feature, $\texttt{pD}_{UXAware}$ exploits a pair of two different δ's, $(\delta_{on}, \delta_{off})$, where δ_{on} is used for screen-on periods while δ_{off} is used for screen-off periods.

Fig. 21 presents our evaluation result of $\texttt{pD}_{UXAware}$ for different $(\delta_{on}, \delta_{off})$'s. The *Y*-axis denotes the average energy saving for 25 users over when no fast dormancy support is used, and the *X*-axis denotes different δ_{off} settings. Each curve represents the evaluation results for three different δ_{on}'s. The result shows that, since δ_{off} can be set as a higher value than δ_{on}, $\texttt{pD}_{UXAware}$ can reduce more radio energy than the original pD with a single δ. For example, when $(\delta_{on}, \delta_{off})$ is set to $(0.10, 0.30)$, $\texttt{pD}_{UXAware}$ saves radio energy on average by 28 percent while pD with $\delta = 0.10$ reduces about 23 percent of the radio energy over the baseline. Furthermore, the result shows that $\texttt{pD}_{UXAware}$ can reduce the radio energy consumption with little negative impact on user experience. For example, $\texttt{pD}_{UXAware}$ achieves about 24 percent of radio energy saving with $(\delta_{on}, \delta_{off}) = (0.01, 0.30)$. Since $\delta_{on} = 0.01$ may introduce up to 1 percent of radio reconnection increase, we can expect that user hardly recognizes any negative change in interacting with a radio network over the case when no fast dormancy feature was used.

6 Related Work

Motivated by the observation that a large amount of smartphone energy is wasted in the tail time [8], [9], several techniques were proposed to save the wasted tail energy. For example, TailEnder [8] delays actual network data transfers if an app can be tolerant of delays in data transmissions. By sending delayed multiple transfers in a batch mode, multiple tail periods between small successive data transfers can be avoided, thus saving the wasted energy in the tail time. TOP [4] suggested an app-level protocol with which apps can provide hints on their future network transmissions to their system. The fast dormancy feature can be invoked if their hints indicate that there is no transmission in the tail time. Both TailEnder and TOP, however, are not applicable for most apps which usually do not provide explicit hints on their network transmissions, thus limiting their effectiveness to a small number of specially designed apps. The proposed pD, on the other hand, has no such restriction on applicable apps, saving the radio energy from most apps.

Recent investigations have more directly focused on how automatically the radio energy can be saved by managing the tail time. For example, RadioJockey [5] predicts the end of transmissions for apps running in the background by monitoring system call traces. When a particular sequence of system calls is predicted to be a meaningful hint for invoking the fast dormancy feature, RadioJockey invokes the fast dormancy feature. Although RadioJockey works well for background tasks whose transmission patterns are user-independent, it cannot be applied for interactive apps which form a majority of apps used by smartphone users. Our proposed pD is fundamentally different in that pD can be applied for interactive apps as well. Furthermore, pD takes advantages of personalized transmission characteristics for users' behaviors in predicting more general transmission patterns.

Deng and Balakrishnan [13] also proposed a predictive technique which determines when to change the radio states. Since they assumed that a recent transmission trend will be maintained in predicting immediate network transmissions, their prediction could be accurate only when the temporal traffic patterns remain unchanged. On the other hand, the proposed pD can make accurate prediction even when different network activities are occurring with different tail-time transmission trends, because pD leverages the distinct characteristics for each network activity. Huang et al. [14] suggested a radio resource management technique for network transmissions when a display screen is off. Our approach is more general in that pD can be applicable screen-on transmissions as well as screen-off transmissions.

7 Conclusion and Future Work

We presented a novel personalized network activity-aware predictive dormancy technique, pD, for reducing the wasted radio energy of smartphones. The pD technique is motivated by a personalized nature of smartphone usage (as validated from our smartphone user study). In particular, we observed that there are strong per-user tail-time transmission patterns for different network activities. Based on this observation, pD automatically extracts network activities from apps (without any extra requirement) and systematically builds tail-time transmission models for the apps. In order to invoke the fast dormancy feature in an effective fashion, pD takes advantage of personalized tail-time network transmission models in predicting when a next transmission will occur in the tail time. Our experimental results validate that pD is an effective solution in reducing the amount of wasted tail energy. pD can save the radio energy consumption on average by 23 percent over when no fast dormancy feature is used with less than 10 percent reconnection increase.

Our current pD technique can be further improved in several directions. For example, if we can distinguish latency-sensitive transmissions from latency-insensitive transmissions by exploiting other context information, pD can more aggressively invoke the fast dormancy feature for latency-insensitive radio transmissions. Our immediate

future work includes building a general framework for identifying different types of data transmissions automatically based on user-centric response time analysis. As a longer-term direction, we plan to investigate if our personalized network activity-aware approach can be extended for other types of system optimizations. For example, it may be possible to reduce user-perceived radio delays when connecting to a mobile network by an intelligent prediction of future network workload.

Acknowledgments

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT and Future Planning (MSIP) (No. 2010-0020724). The ICT at Seoul National University and IDEC provided research facilities for this study.

References

[1] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Conf. USENIX Annu. Tech. Conf.*, 2010, p. 21.

[2] A. Shye, B. Scholbrock, and G. Memik, "Into the wild: Studying real user activity patterns to guide power optimizations for mobile architectures," in *Proc. 42nd Annu. IEEE/ACM Int. Symp. Microarchit.*, 2009, pp. 168–178.

[3] The 3rd Generation Partnership Project, "Configuration of fast dormancy in release 8. 3GPP discussion and decision notes RP-090960"

[4] F. Qian, Z. Wang, A. Gerber, Z.M. Mao, S. Sen, and O. Spatscheck, "TOP: Tail optimization protocol for cellular radio resource allocation," in *Proc. 18th IEEE Int. Conf. Netw. Protocols*, 2010, pp. 285–294.

[5] P. K. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. N. Padmanabhan, and G. Varghese, "RadioJockey: Mining program execution to optimize cellular radio usage," in *Proc. 18th Annu. Int. Conf. Mobile Comput. Netw.*, 2012, pp. 101–112.

[6] C. Gniady, A. R. Butt, and Y. C. Hu, "Program-counter-based pattern classification in buffer caching," in *Proc. 6th Symp. Oper. Syst. Design Implementation*, 2004, p. 27.

[7] Y. Kim and J. Kim, "Personalized diapause: Reducing radio energy consumption of smartphones by network-context aware dormancy predictions," in *Proc. Workshop Power Aware Comput. Syst.*, 2012, p. 1.

[8] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: A measurement study and implications for network application," in *Proc. 9th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2009, pp. 280–293.

[9] F. Qian, Z. Wang, A. Gerber, Z.M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3G networks," in *Proc. 10th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2010, pp. 137–150.

[10] 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 *Proc. 10th Int. Conf. Mobile Syst., Appl., Services*, 2012, pp. 225–238.

[11] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate online power estimation and automatic battery behavior based power model generation for smartphones," in *Proc. 8th Int. Conf. Hardware/Softw. Codesign Syst. Synthesis*, 2010, pp. 105–114.

[12] H. Falaki, R. Mahajan, S. Kandula, D. Lymberopoulos, R. Govindan, and D. Estrin, "Diversity in smartphone usage," in *Proc. 8th Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 179–194.

[13] S. Deng and H. Balakrishnan, "Traffic-aware techniques to reduce 3G/LTE wireless energy consumption," in *Proc. 8th Int. Conf. Emerging Netw. Experiments Technol.*, 2012, pp. 181–192.

[14] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck, "Screen-off traffic characterization and optimization in 3G/4G networks," in *Proc. 12th ACM SIGCOMM Conf. Internet Meas. Conf.*, 2012, pp. 357–364.

[15] T. M. T. Do, J. Blom, and D. Gatica-Perez, "Smartphone usage in the wild: A large-scale analysis of applications and context," in *Proc. 13th Int. Conf. Multimodal Interfaces*, 2011, pp. 353–360.

[16] A. Shye, B. Scholbrok, G. Memik, and P. A. Dinda, "Characterizing and modeling user activity on smartphones: Summary," in *Proc. ACM SIGMETRICS Int. Conf. Meas. Model. Comput. Syst.*, 2010, pp. 375–376.

Yeseong Kim received the BS degree in computer science and engineering from Seoul National University, Seoul, Korea, in 2011. He is currently working toward the PhD degree in the Department of Computer Science and Engineering, University of California at San Diego, La Jolla, CA. His research interests include mobile systems, operating systems, and embedded software.

Boyeong Jeon received the BS and MS degrees from Seoul National University, Seoul, in 2012 and 2014, respectively, all in computer science and engineering. She is currently with the SAP Labs Korea. Her current research interests include embedded softwares, mobile system, and data mining.

Jihong Kim received the BS degree in computer science and statistics from Seoul National University, Seoul, Korea, in 1986, and the MS and PhD degrees in computer science and engineering from the University of Washington, Seattle, WA, in 1988 and 1995, respectively. Before joining SNU in 1997, he was a member of Technical Staff in the DSPS R&D Center of Texas Instruments in Dallas, Texas. He is currently a professor in the School of Computer Science and Engineering, Seoul National University. His research interests include embedded software, low-power systems, computer architecture, and storage systems. He is a member of the IEEE.