Personalized Diapause: Reducing Radio Energy Consumption of Smartphones by Network-Context Aware Dormancy Predictions

Yeseong Kim   Jihong Kim
Department of Computer Science and Engineering
Seoul National University
{*yeseong, jihong*}*@davinci.snu.ac.kr*

Abstract
A large portion of radio energy in smartphones is wasted during a special waiting period, known as the *tail time*, after a transmission is completed. In order to save the wasted energy during the tail time, it is important to accurately predict whether a subsequent transmission will occur in the tail period. In this paper, we propose a novel general-purpose predictive dormancy technique, called Personalized Diapause (PD). By automatically extracting meaningful network activities as network contexts, our proposed technique takes advantage of per-user usage characteristics of each network context in deciding when to release a radio connection within the tail time. Our experimental results using real network usage logs from 25 users show that PD can save the radio energy consumption by up to 36% with about 10% reconnection increase.

1 Introduction

The radio energy consumption in smartphones is steadily increasing. For example, in 3G network smartphones, radio communications are responsible for about 30% of the total energy consumption in smartphones. A significant portion of the high radio energy consumption comes from the energy wasted during a special interval, known as *the tail time*. The tail time refers a fixed-length interval I_{tail} after a packet transmission is completed. During this interval of the length T_{tail}, a radio connection is maintained at high power level. Since re-establishing a radio connection after releasing the radio resource incurs a long delay and a high signaling overhead, the 3G protocol maintains the radio connection during the tail time, expecting that a subsequent transmission is very likely to happen in the tail time. If there is no transmission during I_{tail}, however, a large amount of radio energy is wasted. For example, in our network usage study of 25 Android smartphone users, we observed that, on average, about one third of the total radio energy was wasted during tail times, waiting for a subsequent transmissions (which didn't occur). Since 4G wireless communication standards such as 4G LTE also employ similar tail times, saving wasted energy in tail times are very important in achieving a high energy efficiency in smartphones.

In order to save the wasted energy during the tail time, the fast dormancy feature [1] was recently proposed. The fast dormancy protocol enables a smartphone radio module to quickly release its radio connection *even in tail times* if the radio module decides that no additional data transmission occurs within the tail time. By exploiting the fast dormancy protocol when there is no more subsequent transmission, a smartphone can reduce the energy consumed in the tail time. In utilizing the fast dormancy feature efficiently, a key challenge is to predict whether (or when) a subsequent data transmission will occur in the tail time, after the current data transmission has been completed. If the request time of the next network transmission is mispredicted to occur in the tail time, a large amount of energy is wasted in the tail time. If the next request is mispredicted not to occur in the tail time, a large radio reconnection overhead (both to a smartphone and a mobile network) should be paid.

Existing predictive dormancy techniques such as TOP [2] are, however, difficult to apply for many existing apps because these techniques require some run-time hints from apps. For example, TOP relies on apps for providing hints on the next transmission so that it can decide if a radio connection should be released or not after the current transmission is completed. A multimedia streaming app, for example, may easily provide such hints when a multimedia download is completed while playing the downloaded content, because a user may not need to access a mobile network for a while. Although these techniques can work well when such hints on the next transmission are explicitly provided, most interactive apps such as SNS apps (e.g., a google talk app and a facebook app) cannot accurately estimate the next transmission because it is very difficult to predict how an individual user will interact with the apps. Therefore, the existing techniques are not applicable to apps with more general network transmission patterns. Furthermore, if app developers do not provide such hints, it is very difficult to apply these techniques. Since they rarely pay attention to their apps' transmission behavior, we believe that only a small number of apps can exploit the fast dormancy feature efficiently, thus wasting a significant amount of energy unnecessarily by missing many potential opportunities for exploiting the fast dormancy feature.

Intuitively, what the existing techniques lack is a *systematic* and *automatic* way of extracting meaningful user-level network activities from a running app, not depending on app-assisted future network usage hints from app developers. If such network activities can be automatically identified by a system software, and the system software can accurately estimate future radio communication patterns, the energy efficiency of the radio communication can be significantly improved by exploiting the fast dormancy feature for most existing apps in a more efficient fashion.

In this paper, we propose a novel network activity extraction technique that automatically classifies semantically equivalent network activities. Our technique, which exploits program contexts [3], partitions an app's network activities into a small number of equivalent network contexts. Our Android smartphone user study shows that each extracted network context has unique characteristics for transmission trend in the tail time, and different users behave quite differently even for the same network context. 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 such a novel general-purpose network energy optimization technique, called Personalized Diapause (PD), based on our automatic network context extraction technique. In order to evaluate our proposed PD technique, PD was implemented on Android 2.3 (Gingerbread) smartphones. Our experimental results show that PD can save the radio energy consumption by up to 36% with about 10% increase in the radio reconnection overhead over when no fast dormancy feature is used.

The rest of the paper is organized as follows. We explain our proposed network context extraction technique in Sec. 2. In Sec. 3, we summarize key observations from our smartphone user study on network usages where a network context was used as a basic monitoring unit. We describe the main modules of the proposed PD technique in Sec. 4. Experimental results are reported in Sec. 5. Sec. 6 concludes with a summary and future work.

2 Extraction of Network Context

Atomic Network Transmission In order to group a series of inter-related network transmissions into a meaningful network activity, we first 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, we associate each ANT with its unique ID, called as ANT-ID. ANT-ID $\pi(\tau_i)$ of an ANT τ_i is computed by summing the addresses of functions in the call stack [3] (within the Dalvik VM) that lead to the socket API function that initiates the corresponding ANT.

Network Contexts and Equivalent Network Context Block Using ANTs defined above, we represent the network transmissions of an app A as a sequence $\mathcal{S}_A$ of ANTs, 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 $i < j$. Given the sequence $\mathcal{S}_A$, we construct a sequence $\mathcal{C}_A$ of network con-

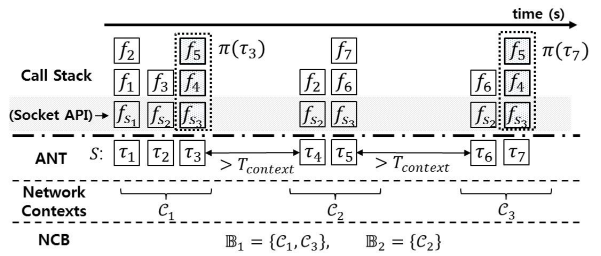

time (s)
Call Stack
(Socket API)
f_2 f_5 $\pi(\tau_3)$ f_7 f_5 $\pi(\tau_7)$
f_1 f_3 f_4 f_2 f_6 f_6 f_4
f_{s_1} f_{s_2} f_{s_3} f_{s_2} f_{s_3} f_{s_2} f_{s_3}
ANT
S: τ_1 τ_2 τ_3 $> T_{context}$ τ_4 τ_5 $> T_{context}$ τ_6 τ_7
Network Contexts
$\mathcal{C}_1$ $\mathcal{C}_2$ $\mathcal{C}_3$
NCB
$\mathbb{B}_1 = \{\mathcal{C}_1, \mathcal{C}_3\}$, $\mathbb{B}_2 = \{\mathcal{C}_2\}$

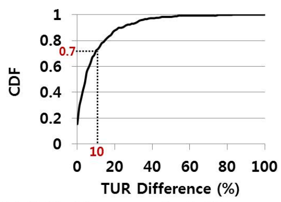
Fig. 1: An example of extracting network context blocks.

texts, $\mathcal{C}_{\mathcal{A}} = \langle \mathcal{C}_1^A, \ldots, \mathcal{C}_k^A \rangle$, where $\mathcal{C}_i^A$ is a network context. A network context $\mathcal{C}_i^A$ consists of successive ANTs $\langle \tau_{i_1}, \tau_{i_1+1}, \ldots, \tau_{i_1+k-1} \rangle$ where the inter-ANT interval between two consecutive ANTs is less than a threshold time $T_{context}$. Therefore, there is at least $T_{context}$ idle transmission interval between any two $\mathcal{C}_i^A$ and $\mathcal{C}_j^A$.

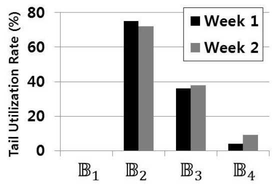
Intuitively, each network context represents a meaningful clustered network activity such as an activity of downloading a song. We define two network contexts, $\mathcal{C}_i^A$ and $\mathcal{C}_j^A$, are equivalent if at least one ANT-ID $\pi(\tau_{i_p})$ of τ_{i_p} in $\mathcal{C}_i^A$ is equivalent 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* (NCB). Each network context in the same NCB represents a semantically similar network activity.

Fig. 1 illustrates how NCBs are automatically extracted using an example. Given a sequence of ANTs $\mathcal{S} = \langle \tau_1, \ldots, \tau_7 \rangle$, 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 to streaming music, $\mathcal{C}_3$ would be assumed to do the same streaming activity. Since the network contexts in the same NCB are assumed to perform the semantically same network activity, we use an NCB as a basic unit of monitoring each user's network activity characteristics.

Immediate Successor of Network Context and NCB We also define the immediate successor context for a network context. For a network context $\mathcal{C}_i$, if $\mathcal{C}_j$ happens after $\mathcal{C}_i$ and there is no other network context between $\mathcal{C}_i$ and $\mathcal{C}_j$, we call $\mathcal{C}_j$ the *immediate successor context* of $\mathcal{C}_i$. In particular, we define the first ANT of the immediate successor context of a network context as the *immediate successor transmission* of a network context. In Fig. 1, 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 $\mathbb{B} = \{\mathcal{C}_1, ..., \mathcal{C}_l\}$, we define the immediate successor transmissions of $\mathbb{B}$ as the set of the immediate successor transmission of each $\mathcal{C}_i \in \mathbb{B}$. (In the rest of this paper, for a network context or an NCB, when there is no confusion, we use 'immediate successor (*IS*)' and 'immediate successor transmission' interchangeably.)



(a) TUR differences for 2,011 combinations of [user, NCB] over 2 weeks.



(b) Per-NCB tail utilization rate.

Fig. 2: Analysis of Tail Utilization Rates.

3 Smartphone Network Usage Analysis

In order to understand how smartphone users interact with a mobile network, we collected detailed ANT logs of smartphone network usage from 25 active smartphone users in Seoul. The study participants represented diverse user groups, aged between 20 and 40, including college students, graduate students, bankers and kindergarten teachers. For this study, we distributed a modified Dalvik VM to the subjects and ANT logs over a period of 2 weeks have been collected. From the collected ANT log of each participant, we extracted NCBs, resulting in 2,011 different [user, NCB] combinations.

In order to evaluate if NCBs are good monitoring units in understanding user's network usage characteristics, we have computed the tail utilization rate (TUR) for each [user, NCB] combination. For a given user u and an NCB $\mathbb{B}$, TUR$(u,\mathbb{B})$ is computed by a ratio of the number of *IS*'s of $\mathbb{B}$ occurred in tail times over the total number of network context invocations of $\mathbb{B}$ in the user u's ANT log. In order to verify whether users tend to react in a similar fashion to the same network activity, we compared two weekly TURs[1] for the same [user, NCB] combination. The result, summarized in Fig. 2(a), shows that for over 70% of 2,011 [user, NCB] combinations, the TUR difference between two weekly TURs is less than 10%. Small TUR fluctuations for the same [user, NCB] combinations strongly suggest that the proposed network context is appropriate in capturing semantically meaningful network activities and users' network transmission tendency.

Although the TUR difference is quite small for a given [user, NCB] combination, TUR values for different NCBs significantly vary even for the same user, as shown in Fig. 2(b). For example, TUR of NCB $\mathbb{B}_1$ is almost zero (i.e., almost no *IS* occurs in the tail time.), thus the tail time is unnecessary for such NCBs. On the other hand, when NCB $\mathbb{B}_2$ is completed, it is likely that an *IS* occurs in the tail time. From this analysis result, we observed that it is important to develop a predictive

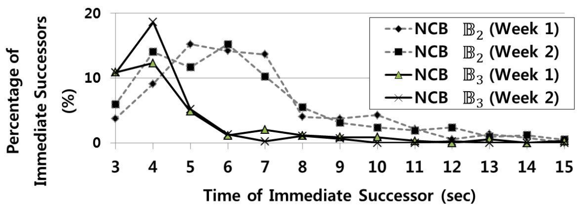


Fig. 3: Distributions of immediate successors in tail times for NCBs $\mathbb{B}_2$ and $\mathbb{B}_3$.

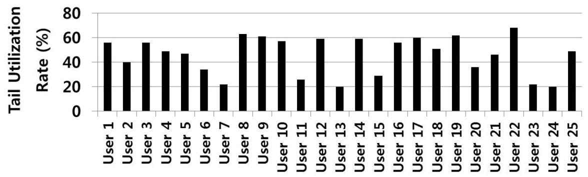


Fig. 4: TUR variations over different users for the same NCB.

dormancy technique that can adapt to user's varying network transmission behavior over different NCBs.

We have also observed that even when TUR for a given [user, NCB] combination is high, actual distributions of *IS* occurrences are quite skewed. Fig. 3 illustrates this observation using *IS* occurrences in the tail times of NCBs $\mathbb{B}_2$ and $\mathbb{B}_3$. Although both $\mathbb{B}_2$ and $\mathbb{B}_3$ have high TUR values as described above (Fig. 2(b)), the *IS* distributions are skewed to the right. For example, most *IS*'s of NCB $\mathbb{B}_3$ happen within the first 5 seconds of the tail time while most *IS*'s of NCB $\mathbb{B}_2$ occur within the first 8 seconds. Fig. 3 also shows that the *IS* skewness of a given NCB is preserved over week-by-week comparisons. Our PD technique exploits these persistent right-skewed distributions in determining the likelihood of an *IS* occurrence, for example, after x seconds in the tail time.

Another important observation was that, even for the same NCB, there is a strong personalized tendency on network transmissions in tail times. As shown in Fig. 4, which shows TURs of the same NCB from a messenger app for 25 users, we observed that TURs for the same NCB are significantly different among different users. For instance, user 22 tends to check his/her messages frequently and react to them quickly, while 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 *IS*'s.

4 Personalized Diapause Architecture

Based on the NCB characteristics discussed in Sec. 3, the proposed PD technique keeps track of TURs for extracted NCBs and decides if an *IS* will occur in the tail time using TUR distributions. Fig. 5 shows an architectural overview of the PD technique. The personalized network activity predictor, which was added as an additional module to the Dalvik VM, is responsible for implementing the PD technique. Whenever the call stack tracer identifies an ANT, the ANT is sent to the network context block extractor module (whose key steps were described in Sec. 2) where related ANTs are grouped into an NCB. Then, the immediate-successor trainer module builds an *IS* model for each NCB. Based on the immediate-successor model, the cost-benefit analysis 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.

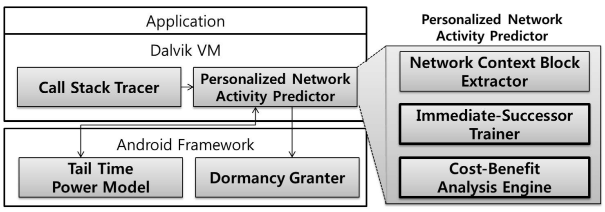

Application
Dalvik VM
Call Stack Tracer
Personalized Network Activity Predictor
Android Framework
Tail Time Power Model
Dormancy Granter
Personalized Network Activity Predictor
Network Context Block Extractor
Immediate-Successor Trainer
Cost-Benefit Analysis Engine

Fig. 5: An architectural overview of Personalized Diapause.

4.1 Immediate-Successor Trainer

In order to predict whether an *IS* will occur or not in the tail time after a given network context of an NCB is completed, PD builds an *immediate-successor model* for each NCB in its immediate-successor trainer module. The key step of the immediate-successor trainer module is to construct a skewed TUR distribution for each NCB.

The immediate-successor model for an NCB $\mathbb{B}$ maintains how many times *IS*'s occur in tail times over the total number of $\mathbb{B}$'s invocations. We divide the inter-context interval between two network contexts into $N+1$ *subintervals*, $s_0, \ldots, s_N$, where each subinterval $s_i = [b_i, e_i)$ can be specified using $b_i = i \cdot \frac{T_{tail}}{N}$, $e_i = (i+1) \cdot \frac{T_{tail}}{N}$ (when $i < N$) and $e_i = \infty$ (when $i = N$). Each subinterval s_i for $i < N$ keeps track of the number n_i of *IS*'s occurred in I_{tail} of $[b_i, e_i)$. For *IS*'s occurred after T_{tail}, we accumulate their occurrences in n_N.

Fig. 6 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 $\mathbb{B}_1$ while $\mathcal{C}_3$ belongs to $\mathbb{B}_2$. 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 after $\mathcal{C}_1$ has been completed.), it increments n_4. For the inter-context interval θ_2, n_7 is incremented since $\theta_2 > T_{tail}$.

4.2 Cost-Benefit Analysis Engine

In order to determine when to invoke the fast dormancy feature, the cost-benefit analysis engine of PD considers the cost-benefit tradeoff based on the immediate-successor model of NCB $\mathbb{B}$. The benefit β_i of $s_i = [b_i, e_i)$, which indicates the expected energy benefit when a radio connection is released at b_i, is defined as $\beta_i = P_{tail} \times (T_{tail} - b_i)$ where P_{tail} is the power consumption within the tail time[2]. The cost C_i of s_i, which indicates the energy penalty when a radio connection needs to be re-established within s_i, is given by $C_i = E_{ohd} + (T_{tail} - b_i) \times P_{tail}$ where E_{ohd} is the energy overhead parameter of re-establishing a radio connection. The second term in C_i is necessary because, once a radio connection is re-established within s_i, the expected energy benefit β_i should be canceled. The gain G_i of s_i, which indicates the energy gain when a radio connection is released at b_i, is defined as $G_i = \beta_i - \sum_{k=i}^{N-1} p_k \times C_k$ where p_k, given by $n_k / \sum_{j=0}^{N} n_j$, represents the probability of *IS* occurrences in s_k. Since G_i assumes that there will be no *IS* in $[b_0, e_{i-1})$, the cost-benefit analysis module actually chooses b_m as the time to invoke the fast dormancy feature where b_m maximizes $(1 - O_{m-1}) \times G_m$

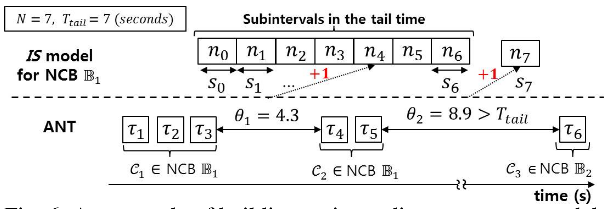

$N = 7,\ T_{tail} = 7\ (seconds)$
Subintervals in the tail time
IS model for NCB $\mathbb{B}_1$
n_0 n_1 n_2 n_3 n_4 n_5 n_6 n_7
s_0 s_1 ... s_6 s_7
+1
+1
ANT
τ_1 τ_2 τ_3 τ_4 τ_5 τ_6
$\theta_1 = 4.3$
$\theta_2 = 8.9 > T_{tail}$
$\mathcal{C}_1 \in$ NCB $\mathbb{B}_1$
$\mathcal{C}_2 \in$ NCB $\mathbb{B}_1$
$\mathcal{C}_3 \in$ NCB $\mathbb{B}_2$
time (s)

Fig. 6: An example of building an immediate-successor model.

where $O_{m-1} = \sum_{j=0}^{m-1} p_j$ $(1 \le m < N)$ and 0 $(m = 0)$. For $1 \le m < N$, O_{m-1} indicates the probability of *IS* occurrences in $[b_0, e_{i-1})$. If multiple apps are using the current radio connection, the dormancy granter first checks if it will be safe for these apps to disconnect the connection.

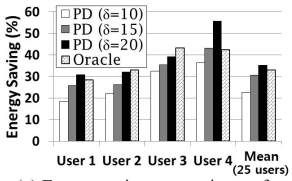
Adaptive Cost-Benefit Tradeoff From the viewpoint of energy saving in *smartphones*, when E_{ohd} is small, it is more beneficial for PD to switch to the dormancy mode more aggressively. However, if E_{ohd} is too small, the frequency of switching to the dormancy mode can be too high. Since too frequent switches can incur a high signaling overhead to a mobile network (as well as longer delays to mobile users), we manage the frequency of dormancy mode switches adaptively by using the soft upper bound on the acceptable number of mode changes. In the current implementation, PD allows about $\delta\%$ increase in the number of reconnections over when no fast dormancy feature is used. For example, if the current number of radio reconnections exceed more than $\delta\%$ of the radio reconnections under no fast dormancy feature, The cost-benefit analysis engine increases E_{ohd} by ΔE_{ohd}, thus making it less likely to release a radio connection. The initial value of E_{ohd} was determined by measurements.

5 Experimental Results

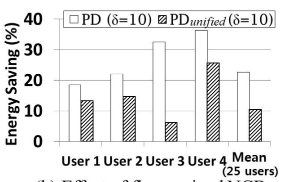
In order to evaluate the efficiency of the proposed PD technique, we have implemented PD on Nexus S Android reference smartphones running Android 2.3 (Gingerbread). We modified Dalvik VM for tracking call stacks that lead to the socket API functions. The additional PD modules which are described in Sec. 4 were also implemented to Dalvik VM and Android framework.

In our experiments, we have used a custom ANT log replayer tool for reproducing the collected ANT logs from 25 users. For energy consumption comparisons, we have used our 3G energy simulator, which was developed based on our smartphone power measurement study in a similar fashion to one used in [4]. The tail time T_{tail} was set to 15 seconds and its power consumption P_{tail} was assumed to be 410 mW. N was set to 15 and $T_{context}$ was set to 3 seconds[3]. The energy overhead of running the PD modules was not included in the presented results, because the PD's impact on the execution time was very small. For example, each NCB extraction and subsequent computations took less than 1 ms. Since the average number of NCBs per app was 3, this extra execution time was negligible.

Fig. 7 shows the impact of PD on the energy con-



(a) Energy saving comparisons of PD with different δ's and Oracle.



(b) Effect of fine-grained NCB groupings on energy saving.

Fig. 7: PD's effect on energy saving and reconnection overhead.

sumption and radio reconnection overhead for four representative users. Using no-fast-dormancy support as a baseline, Fig. 7(a) shows how PD's energy saving ratios change over different δ's. PD was evaluated under the assumption that the increase in the number of radio reconnections is limited by 10%, 15% and 20%, respectively (i.e., $\delta = 10, 15$ and 20). The result shows that PD can save the radio energy on average by 23% over the no-fast-dormancy case with $\delta = 10$. For User 4, the maximum energy saving of 36% is achieved over the no-fast-dormancy case. For $\delta = 15$ and $\delta = 20$, PD saves more energy, on average by 31% and 35% over the base line, respectively.

In order to better understand the energy efficiency of PD, we have also compared PD with the off-line optimal technique, called `Oracle`, which uses an oracle predictor on future network usages. Since `Oracle` has a complete knowledge on future network usages, it achieves the possible maximum energy saving *if no reconnection increase is allowed.* `Oracle`, which is not implementable in practice, is useful in objectively understanding the efficiency of PD. As shown in Fig. 7(a), PD performs close to `Oracle` when $\delta = 15$ and $\delta = 20$. Note that PD saves even more energy than `Oracle` in Users 1 and 4 (as well as the average cases) when $\delta = 20$. This is because, with $\delta = 20$, PD gets more aggressive in disconnecting radio connections in the tail time, at the expense of increased reconnections. The energy efficiency gap between PD and `Oracle` comes mainly from when PD chooses the dormancy mode switch time *in the latter part* of the tail time based on its cost-benefit analysis model while `Oracle` can choose the dormancy mode switch time without this waiting time. Since any practical on-line technique cannot avoid this initial waiting time, although we need more thorough evaluations, we think that PD is a competitive solution among practical on-line techniques.

In order to understand the impact of our proposed fine-grained NCB classification technique on radio energy savings, we compared PD with PD$_{unified}$, a simplified version of PD. PD$_{unified}$ assumes that all NCBs have the same single unified immediate-successor model. That is, the cost-benefit analysis module of PD$_{unified}$ makes mode switch decisions based on one unified immediate-successor model which was constructed over all NCBs. Fig. 7(b) shows that PD achieves on average 12% higher energy saving over PD$_{unified}$ with $\delta = 10$. For User 3, PD saves 26% more radio energy than PD$_{unified}$. This comparison clearly shows that a fine-grained NCB separation based on semantical differences is important in achieving a high energy efficiency.

6 Conclusions

We have presented a new general-purpose predictive dormancy technique, PD, for optimizing the radio energy consumption of smartphones with the fast dormancy feature. Based on a novel automatic extraction technique of meaningful network activities into network context blocks, PD takes advantage of personalized network context usages in deciding when to release a radio connection within the tail time. Our experimental results show that PD can save the radio energy consumption on average by 23% over when no fast dormancy feature is used when 10% reconnection increase is allowed.

Our current work can be extended in several directions. For example, our current definition of equivalent network contexts may be too relaxing. A tighter equivalency definition may be more efficient in finding more meaningful NCBs. As a longer-term direction, we plan to investigate if our 'network context' idea can be extended for other types of system optimizations. For example, we plan to investigate if other useful information can be collected at NCBs which can provide useful hints for various network energy/performance optimizations.

Acknowledgment

This work was supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MEST) (No. R33-10095, and No. 2012-0006417). This work was also supported by the Brain Korea 21 Project in 2012. The ICT at Seoul National University and IDEC provided research facilities for this study.

References

[1] The 3rd Generation Partnership Project, "Configuration of fast dormancy in release 8. 3GPP discussion and decision notes RP-090960," http://www.3gpp.org/ftp/tsg_ran/tsg_ran/TSGR_45/Documents, 2009.

[2] 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. of the 18th IEEE International Conference on Network Protocols, 2010.

[3] C. Gniady, A. R. Butt, and Y. C. Hu, "Program-counter-based pattern classification in buffer caching," in Proc. of the 6th Symposium on Opearting Systems Design and Implementation, 2004.

[4] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy consumption in mobile phones: a measurement study and implications for network application," in Proc. of the 9th ACM SIGCOMM Conference on Internet Measurement Conference, 2009.

Notes

[1]Week-by-week TUR comparisons may not be appropriate if network activities are repeated with a longer period (e.g, two weeks). However, since the collected logs contain 2-week network traces only, we focused on week-by-week comparisons in this paper.

[2]For brevity, we assume that there is one power state in the tail time. If a radio connection is maintained at multiple power states during the tail time (e.g., two different power states in 3G network), such power states can be easily supported.

[3]We evaluated different $T_{context}$ values from 1 sec to 3 sec, but there were only negligible changes in the experimental results.