Screen-Off Traffic Characterization and Optimization in 3G/4G Networks

Junxian Huang
University of Michigan

Feng Qian
University of Michigan

Z. Morley Mao
University of Michigan

Subhabrata Sen
AT&T Labs - Research

Oliver Spatscheck
AT&T Labs - Research

ABSTRACT

Today's cellular systems operate under diverse resource constraints: limited frequency spectrum, network processing capability, and handset battery life. We consider a novel and important factor, handset *screen status*, *i.e.,* whether the screen is on or off, which was ignored by previous approaches for optimizing cellular resource utilization. Based on analyzing real smartphone traffic collected from 20 users over five months, we find that off-screen traffic accounts for 58.5% of the total radio energy consumption although their traffic volume contribution is much smaller. Such unexpected results are attributed to the unique cellular resource management policy that is not well understood by developers, leading to cellular-unfriendly mobile apps. We then make a further step by proposing *screen-aware* optimization, by leveraging the key observation that screen-off traffic is much more delay-tolerant than its screen-on counterpart due to a lack of user interaction. Our proposal can better balance the key tradeoffs in cellular networks. It saves up to 60.92% of the network energy and reduces signaling and delay overhead by 25.33% and 30.59%, respectively.

Categories and Subject Descriptors

C.2.1 [**Network Architecture and Design**]: wireless communication; C.4 [**Performance of Systems**]: design studies, performance attributes

Keywords

Screen-off traffic, cellular network, traffic optimization, fast dormancy, batching, LTE, radio resource optimization

1. INTRODUCTION

Smartphones with cellular data access have become increasingly popular across the globe, with the wide deployment of 3G and emerging LTE [1] networks, and a plethora of applications of all kinds. Cellular networks are typically characterized by limited radio resources and significant device power consumption for network communications. The battery capacity of smartphones cannot be easily improved due to physical constraints in size and weight. Hence, battery life remains a key determinant of end-user experience. Given the limited radio resources in these networks and device battery capacity constraints, optimizing the usage of these resources is critical for cellular carriers and application developers.

In 3G and 4G cellular networks, the user equipment (UE) must stay in a high-power state, occupying radio resources for some required time before the allocated resource is released by the network, and then the UE enters a low power state. This required time period, also known as the Radio Resource Control (RRC) tail [6], is necessary and important for cellular networks to prevent frequent state promotions (resource allocation), which can cause unacceptably long delays for the UE, as well as additional processing overheads for the radio access network [4, 12]. Today's cellular carriers use a static and conservative setting of the tail time in the order of many seconds, and previous studies have revealed this tail time to be the root cause of energy and radio resource inefficiencies in both 3G [15, 6, 11, 7] and 4G networks [9]. Various optimization solutions have been proposed to address this problem, *e.g.,* the use of fast dormancy [2, 3, 16] and client-side traffic shaping and scheduling [14, 18, 6]. In addition, specialized energy saving techniques for mobile applications have been proposed for specific applications [21, 10] and for specific protocols [5].

In this paper we consider a novel angle to the above problem and explore the impact of **screen status**, *i.e.,* whether the screen is on or off, on the device's network traffic patterns. The screen status is easy to monitor for most mobile OSes. We propose that the screen-off traffic should not be treated the same as the screen-on traffic for traffic optimization purposes, and the former can be optimized more aggressively. The main intuition is that the user (and possibly application) behavior have significant differences when the screen is on *v.s.* off, resulting in different traffic patterns and different performance requirements. When the screen is off, there is a much higher chance that the user is not actively interacting with the device and the network traffic is most likely to be more delay tolerant. Hence we can be more aggressive in optimizing this traffic using techniques such as batching and fast dormancy. In contrast, when the screen is on, it is harder to predict the delay sensitivity of the network traffic and aggressive optimizations may harm the user experience. To validate this intuition, we characterize the screen-off traffic for a real-world user data set and evaluate the benefits of us-

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. To copy otherwise, to republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee.
IMC'12, November 14–16, 2012, Boston, Massachusetts, USA.
Copyright 2012 ACM 978-1-4503-1705-4/12/11 ...$15.00.

ing "screen-aware" optimization for balancing UE energy savings and the resulting overheads in radio resource usage and response delay.

The proposed screen-aware optimization focuses on the overall device traffic, and is complementary to other efficiency improvement efforts, *e.g.,* better application design. While it is important for individual mobile applications to be better designed, the power state on a device is a function of the traffic pattern across all the running applications. Another advantage is that the approach works across both optimized and unoptimized (*e.g.,* legacy) applications.

The main contributions of this study are as follows:

- We perform the first characterization of screen-off traffic using 20 smartphone users' traffic over a 5 month interval. We compare the differences between screen-off and -on traffic patterns, both overall and for different individual applications.

- We find that although the number of packets and total payload for screen-off traffic are smaller than that of screen-on traffic, the former accounts for a disproportionate majority of the total network energy consumed by a device - 58.55% for all users, and up to 73.53% for some user.

- Evaluating various optimization techniques using an LTE RRC state machine energy model, we find that fast dormancy, for the same parameter settings, generates more energy savings and less signaling/delay overhead for screen-off traffic, compared with that for screen-on traffic. This can be attributed to the different traffic patterns for the two traffic classes. In total, the screen-aware optimization achieves 60.92% energy savings, 25.33% reduction in signaling overhead, and 30.59% reduction in channel scheduling delays, compared with the default smartphone settings.

2. METHODOLOGY

We next describe the data set used in this study, the burst analysis methodology, as well as the network and power simulation methodology.

2.1 UMICH data set

The UMICH data set [9] totals 118 GB and consists of five-month (05/12/2011 $\sim$ 10/12/2011) data (both cellular and WiFi) from 20 volunteers who used smartphones running Android 2.2. There are three types of data: *(i)* packet traces in **`tcpdump`** format including both headers and payload, *(ii)* the process name responsible for sending or receiving each packet, using the method described in [17] by correlating the socket, inode, and process ID in Android in realtime, and *(iii)* screen on/off status data with a sampling rate of 1*Hz*. We strictly follow an anonymous analysis procedure to protect the users' privacy.

In order to associate individual packets with their screen status, we define a time window $[t_1, t_2]$ to be a screen-on (or screen-off, respectively) window if "all" the screen samples in this window have screen-on (or screen-off) status. Then we classify a packet as either screen-on or off if its timestamp falls respectively into any screen-on or screen-off window, and unknown otherwise. One reason for the occurrence of the unknown category is data collection errors.

2.2 Burst Analysis Methodology

In order to understand the screen-off traffic pattern and its impact on radio resource and device energy, we use the following traffic model for burst analysis. Intuitively, a burst is a continuous data transfer with preceding and succeeding idle times. For each user, the traffic trace is a sequence of network packets, $P_i (1 \leq i \leq n)$. Notice that P_i could be either downlink or uplink. If the timestamp of P_i is defined to be t_i, we have $t_i \leq t_j$ for any $i < j$. Using a burst threshold **`BT`**, the packets are divided into ***bursts***, *i.e.,* $\{P_p, P_{p+1}, \cdots, P_q\}$ belongs to a burst B, if and only if: *i)* $t_{k+1} - t_k \leq$ **`BT`** for any $k \in \{p, \cdots, q-1\}$, *ii)* $t_{q+1} - t_q >$ **`BT`** and *iii)* $t_p - t_{p-1} >$ **`BT`**. We define the inter-burst time **`IBT`** for burst B to be the time gap following this burst, *i.e.,* $t_{q+1} - t_q$. In this paper, we empirically choose to use **`BT`** = 2 seconds, which is validated to be larger than most packet gaps for 3G/4G networks within a continuous data transfer, such as downloading a web object.

2.3 LTE Network and Power Simulation Methodology

For evaluating optimization performance, we rely on network and power simulations. In this paper, we focus on the emerging LTE cellular technology to understand the energy and radio resource overhead of a particular packet trace. However, since there are important similarities in the network and power models for the LTE and 3G UMTS networks [8, 13, 19], *e.g.,* distinct energy states with very different power consumptions, long tail times *etc.*, the overall conclusions apply to 3G as well.

Specifically, we use the LTE 4G network and power simulation model, as well as three important performance metrics E, S and D, detailed in our earlier work [9]. E is the total UE network energy - the energy consumed by the device's cellular network interface. A previous study [20] indicates that this network energy typically accounts for a third of the energy drain on the phone. Given that E is larger when the allocated radio resource is occupied by the UE for a longer duration, reducing E is aligned with reducing the total radio resource occupation time. S is defined to be the signaling overhead, *i.e.,* the number of RRC promotions from **`RRC_IDLE`** (the idle state for LTE) to **`RRC_CONNECTED`** (the high-power state for LTE) triggered by the packets in a specific trace. Such state transitions cause significant signaling loads on the RAN and reducing this overhead is therefore important. D is the user-perceived channel scheduling delay, including promotion delay, caused by state promotions from **`RRC_IDLE`** to **`RRC_CONNECTED`**, and paging delay caused by UE's sleeping before checking the data channel (a mechanism in the cellular network to save energy). Reducing D is desirable as it improves user experience.

2.4 Traffic Optimization Methodology

In this paper, we mainly focus on two traffic optimization techniques:

(i) *Fast dormancy (FD)* [2, 3] is a mechanism in 3G networks for reducing the amount of tail time incurred by a device by quickly demoting it to a low energy RRC state without waiting for the tail timer to expire. In our simulations, we explore fast dormancy in an LTE setting, motivated by the fact that LTE, like 3G, also suffers from a serious tail problem [9]. Our fast dormancy-based optimization works as follows: when the UE has not observed any network activity for some idle time T_i, it sends a special RRC message to

	# of packets (million)	%
Total	131.49	100%
Screen on	72.50	55.13%
Screen off	47.14	35.84%
Unknown	11.85	9.02%

Table 1: Packet statistics of the UMICH **data set.**

the network to make the allocated radio resource be released earlier, instead of occupying it for the whole RRC tail. After the radio resource is released, the UE switches to the low power idle state (**`RRC_IDLE`** for LTE networks), saving energy. The setting of T_i is important for balancing the tradeoffs among UE energy saving (ΔE), signaling overhead (ΔS) and channel scheduling delay (ΔD), *i.e.,* a smaller T_i would results in a larger ΔE at the cost of larger ΔS and ΔD, and *vice versa*.

(ii) *Batching* is a widely used traffic shaping technique, which has been discussed in previous studies [6, 14]. In this study, batching uses two parameters, source window size α (seconds) and target window size β (seconds), and $\alpha > \beta$. For each α seconds long time window $[t, t + \alpha]$, packets within $[t, t + \alpha - \beta]$ are delayed and batched with those inside $[t+\alpha-\beta, t+\alpha]$. Notice that if α/β is too large, the limited bandwidth of the cellular network could become the bottleneck, making the batching impossible. So in this study, we make sure that our choice of α and β results in acceptable bandwidth usage for LTE networks.

3. CHARACTERISTICS OF SCREEN-OFF TRAFFIC

In this section, we present packet characteristics and burst analysis of screen-off traffic in the UMICH data set. Then we compare screen-on and screen-off traffic and scrutinize the top applications generating screen-off traffic.

3.1 Packet Characteristics of Screen-off Traffic

Using the methodology described in §2.1, we classify all packets to be screen-on, screen-off or unknown. Table 1 lists the number of packets in different categories. Among the total 131.49 million packets, 55.12% of them are screen-on packets and 35.85% are screen-off packets, with 9.02% unknown. The possible reasons for unknown packets are multifold, including that voluntary users may have accidentally killed the data collector. For the unknown category, we conservatively choose not optimize it for either screen-on or screen-off traffic optimization.

The top section of Table 2 lists the packet characteristics of both screen-on and screen-off traffic. Packet payload size refers to the size in bytes of an IP packet excluding the TCP/UDP and IP headers, and payload for a given process is the sum of the packet payload sizes of all packets corresponding to that process. We notice that screen-off traffic has far less packets (35.84% of total) than screen-on traffic (55.13% of total), much smaller total payload (27.26% for screen-off and 64.31% for screen-on traffic), and smaller average downlink packet payload size.

To understand individual process behavior within screen-off traffic, we scrutinize the top processes sorted by the number of screen-off packets, in the bottom section of Table 2. In the second column, titled "Off payload", we observe that some processes have most of their payload transferred during screen-off sessions, *e.g.,* Genie Widget, Epicurious Recipe, *etc.* Especially for Yahoo! Sportacular, 80.45% of all its payload is transferred when the screen is off. This is possibly due to the background behaviors of these processes, involving either periodically pulling updates from servers or traffic triggered by server-initiated push notifications. In terms of the % of downlink payload, compared to other processes, Skypekit, Gmail, Sportacular and Facebook have smaller proportions of their respective total downlink payloads associated with the screen-off states. These processes also have a smaller average downlink packet size (<500B). In contrast, a process like **`android.process.media`** has an average size of 1388B, indicating that most packets have a size of MTU (maximum transmission unit, ∼1500B). This shows that for screen-off traffic, application behavioral diversity still exists similar to that of screen-on traffic.

A group of processes share quite similar behavior patterns, including Google Music, **`/system/bin/mediaserver`** and **`android.process.media`**, which have larger payload (>2GB), and >99% of the payload is downlink. Their average downlink packet sizes are close to the MTU and average uplink packet sizes are close to 0. Also, their ratio of downlink packets is close to 2/3. This is because TCP's delayed ACK would generate one uplink ACK packet for every two downlink data packets, resulting in a ratio of 2/3 of downlink packets. These observations suggest that Google Music is downloading large amount of data, as it can run in the background allowing users to listen to the music with the screen off. Similarly, although **`/system/bin/mediaserver`** and **`android.process.media`** are not actual applications, they are used by other applications, such as Pandora, to download contents for users while the screen is off. However, this group of processes does not necessarily have higher energy consumption compared with the remaining processes, and we explore this in more detail in latter sections.

3.2 Burst Analysis of Screen-off Traffic

Following the methodology in §2.2, Table 3 lists the results of the burst analysis for screen-off traffic, with that of screen-on traffic listed for comparison purposes.

We observe that screen-off traffic contains much more bursts than screen-on traffic, although the total number of packets for screen-off traffic is smaller. For screen-off traffic, bursts are smaller in terms of the number of downlink/uplink packets and payload. Especially, for average downlink payload per burst, screen-on traffic is 7 times that for screen-off traffic. In addition, the average burst length and the **`IBT`** following bursts for screen-off traffic are both shorter than those of screen-on traffic. The above observations indicate that screen-off bursts are smaller in size and duration and appear more often — such behavior is likely to cause longer channel occupation time in the high energy RRC state and therefore incur significant battery usage.

By studying the screen-off burst behavior of individual processes, we classify them into two separate groups. The first group, which we call *Gathered* group, includes Genie Widget, Google Music, **`/system/bin/mediaserver`** and **`android.process.media`**. These processes have a small number of larger bursts in terms of the number of uplink/downlink packets per burst and the average downlink payload. Notice that the uplink payload for these bursts is not necessarily large, since a small uplink payload of HTTP request can result in a large file download. The *Gathered* group also has longer bursts and longer trailing **`IBT`** in average, indicating a less frequent appearance.

Traffic type	Payload (GB) / %[a]	% of downlink payload	# of packets ($\times 10^6$) / %[b]	% of downlink packets	Avg downlink packet payload size (B)	Avg uplink packet payload size (B)
Screen-on	51.47 / 64.31%	96.31%	72.50 / 55.13%	60.71%	1126	67
Screen-off	21.82 / 27.26%	93.52%	47.14 / 35.84%	52.60%	823	63

Process name	Off payload (GB) / %[c]	% of downlink off payload[d]	# of off packets ($\times 10^6$) / %[e]	% of downlink off packets[f]	Avg downlink off packet payload size (B)	Avg uplink off packet payload size (B)
Genie Widget	1.76 / 72.21%	97.01%	3.80 / 73.16%	49.97%	901	28
Google Music	3.13 / 57.14%	99.91%	3.30 / 57.02%	68.60%	1384	3
Epicurious Recipe	1.65 / 70.05%	99.22%	2.69 / 69.29%	50.46%	1212	10
/system/bin/mediaserver	2.39 / 10.09%	99.77%	2.66 / 11.05%	66.95%	1342	6
android.process.media	2.35 / 28.42%	99.98%	2.37 / 29.06%	71.55%	1388	1
Skypekit[g]	0.04 / 25.54%	48.44%	2.07 / 46.73%	48.32%	22	22
Facebook	0.46 / 32.96%	86.13%	1.95 / 40.67%	42.55%	487	58
Yahoo! Sportacular	0.23 / 80.45%	83.53%	1.94 / 81.05%	41.98%	238	34
Gmail	0.39 / 46.00%	63.65%	1.33 / 54.46%	47.70%	400	208

[a] Payload refers to the total screen-on/off payload, and % is relative to the total payload of all traffic.
[b] % relative to the total number of packets of all traffic.
[c] Off payload refers to the screen-off payload of the specific application, and % is relative to the total payload of this application.
[d] % of downlink screen-off payload of the specific application relative to the total screen-off payload of that application.
[f] % of downlink screen-off packet count of the specific application relative to the total screen-off packet count of that application.
[g] Full process name: /data/data/com.skype.raider/files/skypekit, which is not the actual Skype application (com.skype.raider).

Table 2: Packet characteristics of screen-on/off traffic and top processes for screen-off traffic.

Traffic type	# of bursts	Avg[a] # of uplink packets	Avg[a] # of downlink packets	Avg[a] uplink payload (B)	Avg[a] downlink payload (KB)	Avg[a] burst length (sec)	Avg[a] `IBT` following (sec)
Screen-on	650,941	43.75	67.62	2910.44	76.17	2.92	335.13
Screen-off	1,910,939	11.69	12.98	739.78	10.68	1.37	113.60

Process name	# of bursts	Avg[a] # of uplink packets	Avg[a] # of downlink packets	Avg[a] uplink payload (B)	Avg[a] downlink payload (KB)	Avg[a] burst length (sec)	Avg[a] `IBT` following (sec)
Genie Widget	5,952	319.73	319.36	8852.48	287.88	17.87	3,892.87
Google Music	5,297	195.69	427.56	505.54	591.92	4.53	5,111.50
Epicurious Recipe	63,236	21.07	21.46	202.22	26.01	0.67	159.34
/system/bin/mediaserver	8,163	106.44	215.53	669.82	289.35	5.01	14,451.70
android.process.media	1,442	461.88	1,156.93	246.99	1,605.84	19.83	123,565.00
Skypekit	42,744	25.08	23.46	555.38	0.52	1.93	832.79
Facebook	203,535	5.49	4.07	318.83	1.98	0.86	547.23
Yahoo! Sportacular	133,785	8.39	6.07	285.44	1.45	1.52	261.78
Gmail	105,478	6.60	6.02	1375.30	2.41	1.17	2,002.60

[a] Each "avg" in this table stands for the average value per burst.

Table 3: Burst analysis of screen-on/off traffic and top processes for screen-off traffic.

The rest of the processes fall into the second group, called the *Scattered* group, which generate significantly more bursts and on average, these bursts contain less packets and smaller downlink payload. In addition, these bursts are shorter in duration and appear more frequently. A representative process from this group is Facebook, which includes over 200,000 bursts, and the major reason for this behavior of Facebook is the periodic keep-alive transfers [14].

Based on this comparison, we believe that it is both easy and important for mobile application developers to optimize their application behaviors during the screen-off stage, *e.g.,* for delay-insensitive traffic, they can batch the data into larger bursts, or even eliminate the screen-off data transfers if they are not necessary.

4. RADIO RESOURCE, ENERGY IMPACT AND OPTIMIZATION

Using the network and power model simulation (§2.3), we now evaluate the radio resource and energy impact of screen-off traffic and evaluate some optimization approaches.

4.1 Radio Resource and Energy Impact of Screen-off Traffic

Table 4 presents highlights of the simulation results using the LTE network and power model defined in our previous work [9]. ΔE, ΔS and ΔD represents the change of network energy, signaling overhead and channel scheduling delay after removing the traffic of an application or type, and a negative value indicates a reduction. The results indicate that, compared to screen-on traffic, screen-off traffic clearly has larger impact on the network energy E, as well as S and D. For example, removing all screen-off traffic reduces the total network energy by 58.55%, and for one user, this reduction is as high as 73.53%.

Comparing the *Gathered* group and *Scattered* group discussed in §3.2, the former has very small impact on E, S and D, while the later has a substantial impact. This is because, for the *Scattered* group, a large number of small bursts could result in a large number of RRC tails if the **`IBT`**s among these bursts are larger than the tail time and a long channel occupation time otherwise. For example,

Traffic type	ΔE %[a]	Min $\|\Delta E\|$ %[c]	Max $\|\Delta E\|$ %[c]	ΔS %[a]	ΔD %[a]
Screen-on	-22.18%	3.78%	38.11%	-14.87%	-17.57%
Screen-off	-58.55%	12.39%	73.53%	-58.03%	-54.46%

Process name	ΔE %[b]	Min $\|\Delta E\|$ %[c]	Max $\|\Delta E\|$ %[c]	ΔS %[b]	ΔD %[b]
Genie Widget	-0.34%	0%	2.46%	0.11%	-0.81%
Google Music	-0.12%	0%	1.68%	-0.02%	-0.08%
Epicurious Recipe	-1.63%	0%	26.06%	-1.78%	-0.94%
/system/bin/mediaserver	-0.13%	0%	1.08%	0.02%	-0.16%
android.process.media	-0.08%	0.01%	0.52%	0.02%	-0.05%
Skypekit	-0.96%	0%	7.32%	-0.24%	-0.51%
Facebook	-5.25%	0%	34.68%	-5.82%	-4.15%
Yahoo! Sportacular	-3.01%	0%	20.04%	-1.88%	-1.44%
Gmail	-1.18%	0.04%	4.59%	-0.69%	-1.47%

[a] ΔE, ΔS and ΔD are calculated by removing all screen-on/off traffic from the original traces
[b] ΔE, ΔS and ΔD are calculated by removing the screen-off traffic of one process from the original traces
[c] Min and max refer to the minimum and maximum energy saving $|\Delta E|$ across all users, respectively.

Table 4: Radio resource and energy impact of screen-on/off traffic and top processes for screen-off traffic.

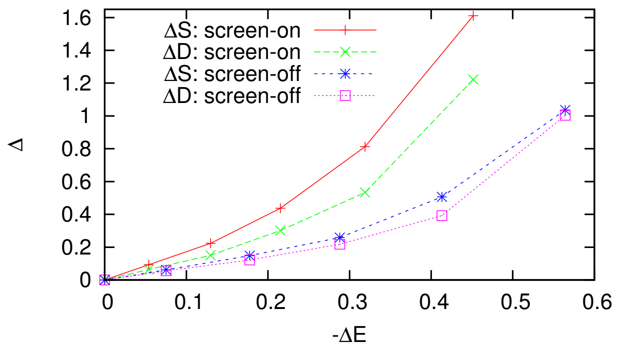

Figure 1: Effectiveness comparison of fast dormancy.

although the Facebook process does not generate the most screen-off traffic, it has the largest energy impact among all processes, *i.e.,* 5.25% of the total network energy can be saved by only removing Facebook's screen-off traffic, and for some users this number could be as high as 34.68%.

4.2 Traffic Optimization

Based on the above analysis, we find that screen-off traffic has a clearly different pattern compared to screen-on traffic, and accounts for a huge proportion of the UE network energy E, signaling overhead S and channel scheduling delay D. At the same time, because screen-off traffic is likely to be more delay-tolerant (see discussions in §1), it is more amenable to more aggressive optimization efforts. Intuitively a traffic optimization approach that is appropriately tuned to the two different traffic categories would yield significant efficiencies. To verify this intuition, we study two common optimization techniques, fast dormancy and batching.

We first compare applying fast dormancy to screen-on and screen-off traffic with a separate T_i settings. The default setting of the **`RRC_CONNECTED`** inactivity timer is 11.58 seconds for a major LTE ISP [9]. In Figure 1, we vary the setting of T_i from 10 seconds to 2 seconds with a step-size of 2 seconds (from left to right in Figure 1) and calculate the $\Delta(E, S, D)$ relative to when fast dormancy is not applied. We observe that for the same T_i setting, there is a higher energy saving ΔE and a lower ΔS and ΔD for screen-off traffic, compared with screen-on traffic, and these gaps are larger when T_i is smaller, *i.e.,* applying fast dormancy more aggressively. So in order to achieve the same energy saving for screen-off and screen-on traffic, the T_i should be set to a much smaller value for screen-on traffic, incurring much larger ΔS and ΔD. However, the responsiveness requirement when the screen is on is actually higher, hence a higher ΔD for screen-on traffic is not acceptable. A better strategy would be being more aggressive (a smaller T_i) for screen-off traffic, which produces significant energy savings, and more conservative (a relatively larger T_i) for screen-on traffic, which limits the negative impact on user experience, though with less energy savings. For example, when $T_i = 8s$, there is 42% energy savings with 52% ΔS and 40% ΔD for screen-off traffic, and when $T_i = 4s$, for screen-on traffic, though the energy saving is small (24%), ΔS (43%) and ΔD (30%) are also limited. With these two fast dormancy settings, for the whole traffic, we can achieve 34% energy saving, with 47.04% ΔS and 35.21% ΔD. This is a better tradeoff than one single T_i setting for both screen-on and screen-off traffic.

Besides fast dormancy, batching [14, 6] is also proposed for optimizing mobile traffic. In Table 5, we compare fast dormancy and batching under different settings for all applications together, and also individually for two applications, Facebook representing the *Scattered* group and Google Music representing the *Gathered* group. For fast dormancy, with reduced E, there is increased S, D, while for batching, all E, S, D are decreased. This is because for fast dormancy, since UE demotes to **`RRC_IDLE`** earlier, there would be more promotions resulting in increased S, D, while for batching, since the traffic pattern is altered, scattered packets are gathered into groups and hence there are less promotions. Notice that the metric D here does not include the delay of packets incurred by batching. In this work, we only focus on batching for screen-off traffic, since any delay for screen-on traffic is likely to be more perceptible to users.

In Table 5, when applying fast dormancy to all applications, we set a different T_i for screen-on/off traffic, *i.e.,* $T_{i,on}$ and $T_{i,off}$. For simplicity, let $<a, b>$ stands for the case when $T_{i,on} = a$ seconds and $T_{i,off} = b$ seconds. Based on Figure 1, we empirically select two values for $T_{i,on}$ and $T_{i,off}$, 4s as an aggressive setting and 8s as a conservative setting. Compared with $<8, 8>$, reducing $T_{i,on}$ to 4s, *i.e.,* $<4, 8>$, only saves 4.21% energy additionally.

Process name	Optimization	Settings	ΔE^a	ΔS^a	ΔD^a
All applications	Fast dormancy	$T_{i,on}$ [b] = 8s, $T_{i,off}$ [b] = 8s	-16.39%	16.95%	13.14%
		$T_{i,on}$ = 4s, $T_{i,off}$ = 8s	-20.60%	28.29%	21.26%
		$T_{i,on}$ = 8s, $T_{i,off}$ = 4s	-34.44%	47.04%	35.21%
		$T_{i,on}$ = 4s, $T_{i,off}$ = 4s	-38.66%	58.38%	43.31%
	Batching	Only for screen-off, α = 50s, β = 10s	-22.33%	-6.24%	-11.27%
		Only for screen-off, α = 50s, β = 5s	-27.15%	-6.24%	-10.67%
		Only for screen-off, α = 100s, β = 10s	-36.72%	-30.00%	-33.43%
		Only for screen-off, α = 100s, β = 5s	-40.79%	-30.00%	-34.25%
	Fast dormancy + Batching	$T_{i,on}$ = 8s, $T_{i,off}$ = 4s, batching only for screen-off traffic, α = 100s, β = 5s	-60.92%	-25.33%	-30.59%
Facebook[c]	Fast dormancy + Batching	$T_{i,on}$ = 8s, $T_{i,off}$ = 4s, batching only for screen-off traffic, α = 100s, β = 5s	-60.19%	-36.27%	-34.93%
Google Music[c]	Fast dormancy + Batching	$T_{i,on}$ = 8s, $T_{i,off}$ = 4s, batching only for screen-off traffic, α = 100s, β = 5s	-57.30%	7.12%	-21.11%

[a] $\Delta E, S, D$ are relative to the E, S, D of all traffic for the specific application.
[b] $T_{i,on}$ is the inactivity threshold of fast dormancy for screen-on traffic, and $T_{i,off}$ is for screen-off traffic.
[c] For these two application rows, we consider the traffic of *only* one specific application, excluding that from other applications.

Table 5: Traffic optimization with fast dormancy and batching.

And $<8, 4>$ has 11.34% reduction in S and 8.10% reduction in D, with only 4.22% less energy saving, compared with $<4, 4>$. This verifies that a different setting for $T_{i,on}$ and $T_{i,off}$ balances the tradeoff of saving energy and reducing overhead, and we select $<8, 4>$ as a reasonable setting. Notice that there are other possible parameter settings representing different aggressiveness with different energy saving and overhead, which may be more appropriate for different settings.

In Table 5, batching applied to the screen-off traffic is able to reduce all E, S, and D. Notice that we do not apply batching for screen-on traffic since it may affect the user experience, *e.g.,* when the user is waiting for a response at real time. We observe that most of the screen-off traffic (in terms of the energy impact) is less delay-sensitive, *e.g.,* push notification, since user interaction is not involved. However, there are also some exceptions for screen-off applications which requires real-time data transfer, *e.g.,* when the user is making a VoIP call with the screen off. Ideally, traffic from these delay-sensitive applications should not be batched even during screen-off stage. In this study, we do not attempt to completely solve this problem, instead, we show an upper bound of the benefit by batching all screen-off traffic. In reality, we need to prioritize delay-sensitive traffic during screen-off stage, and we leave it to future study. In §2.4, we discuss that the choice of α and β values is limited by the available bandwidth for 3G/4G networks. Comparing among the empirically selected candidate settings in Table 5, α = 100s and β = 5s is a better setting, which saves up to 40.79% energy, with a 30.00% reduction in S and 34.25% in D. Notice that the α and β settings studied are just example settings that work well in practice. The goal is to demonstrate the benefit of batching for screen-off traffic and the selection of optimal settings is left to future work.

Then we evaluate applying fast dormancy and batching jointly for all applications and for two sample applications, with the settings specified in Table 5. For all applications, there is a total network energy saving of 60.19%, with 25.33% reduction in S and 30.59% reduction in D. Facebook has similar energy saving, and due to its more "scattered" traffic pattern for screen-off traffic, the batching optimization results in even more reduction for S and D. However, for Google Music, whose traffic is already "gathered" as large bursts, the impact of fast dormancy is more obvious than batching, hence the reduction for D is smaller and there is even an increase in S, unlike the other two scenarios.

5. DISCUSSIONS AND CONCLUSION

In this study, we took a first step towards understanding the impact of screen status on cellular application traffic behavior. Our evaluations in the context of LTE cellular networks, show that although the number of packets and total payload for screen-off traffic are much smaller than that for screen-on traffic, the former accounts for a disproportionate majority (58.55%) of the total network energy consumed by a device. Exploration of resource optimization techniques like fast dormancy and batching indicate that the strategy of optimizing the screen-off traffic more aggressively than screen-on traffic can realize substantial resource savings, without adversely impacting user experience.

We are pursuing this research further, first to explore screen-off traffic in greater detail and explore optimization strategies tailored to the potentially different delay-requirements of subsets of that traffic. Second, in this paper, our optimization strategies imposed the strict constraint that screen-on traffic should not suffer any additional delays that come with traffic shaping approaches. In reality, some limited delay jitter would be tolerable depending on the application and traffic semantics - this is be an additional source of resource optimization beyond the savings we have shown in this paper. Finally, logistics permitting, it would be nice to extend the study to a larger user group.

6. ACKNOWLEDGEMENT

We thank Chen-nee Chuah for her many helpful comments and suggestions for improving the final version of this paper. We also thank the anonymous reviewers for their feed- back and helpful suggestions. This research was partially supported by the National Science Foundation under grants CNS-1111699, CNS-1059372, CNS-0939707, and CNS-1050157.

7. REFERENCES

[1] 3GPP LTE. `http://www.3gpp.org/LTE`.

[2] UE "Fast Dormancy" behavior. 3GPP discussion and decision notes R2-075251, 2007.

[3] Configuration of fast dormancy in release 8. 3GPP discussion and decision notes RP-090960, 2009.

[4] System impact of poor proprietary fast dormancy. 3GPP discussion and decision notes RP-090941, 2009.

[5] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for VoIP over WiFi smartphones. In *MobiSys*, 2007.

[6] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *IMC*, 2009.

[7] M. Chuah, W. Luo, and X. Zhang. Impacts of Inactivity Timer Values on UMTS System Capacity. In *Wireless Communications and Networking Conference*, 2002.

[8] H. Holma and A. Toskala. HSDPA/HSUPA for UMTS: High Speed Radio Access for Mobile Communications. John Wiley and Sons, Inc., 2006.

[9] 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 *MobiSys*, 2012.

[10] S. Kang, J. Lee, H. Jang, H. Lee, Y. Lee, S. Park, T. Park, and J. Song. Seemon: scalable and energy-efficient context monitoring framework for sensor-rich mobile environments. In *MobiSys*, 2008.

[11] C.-C. Lee, J.-H. Yeh, and J.-C. Chen. Impact of inactivity timer on energy consumption in WCDMA and CDMA2000. In *the Third Annual Wireless Telecommunication Symposium (WTS)*, 2004.

[12] P. P. C. Lee, T. Bu, and T. Woo. On the Detection of Signaling DoS Attacks on 3G Wireless Networks. 2007.

[13] J. Perez-Romero, O. Sallent, R. Agusti, and M. Diaz-Guerra. Radio resource management strategies in UMTS. John Wiley and Sons, Inc., 2005.

[14] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Periodic Transfers in Mobile Applications: Network-wide Origin, Impact, and Optimization. In *World Wide Web*, 2012.

[15] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *IMC*, 2010.

[16] 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. ICNP*, 2010.

[17] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Profiling Resource Usage for Mobile Applications: a Cross-layer Approach. In *MobiSys*, 2011.

[18] M. Ra, J. Paek, A. Sharma, R. Govindan, M. Krieger, and M. Neel. Energy-delay tradeoffs in smartphone applications. In *MobiSys*, 2010.

[19] S. Sesia, I. Toufik, and M. Baker. LTE: The UMTS Long Term Evolution From Theory to Practice. John Wiley and Sons, Inc., 2009.

[20] A. Shye, B. Sholbrock, and M. G. Into the wild: Studying real user activity patterns to guide power optimization for mobile architectures. In *Micro*, 2009.

[21] Y. Wang, J. Lin, M. Annavaram, Q. Jacobson, J. Hong, B. Krishnamachari, and N. Sadeh. A framework of energy efficient mobile sensing for automatic user state recognition. In *MobiSys*, 2009.