Periodic Transfers in Mobile Applications: Network-wide Origin, Impact, and Optimization

Feng Qian
University of Michigan

Zhaoguang Wang
University of Michigan

Yudong Gao
University of Michigan

Junxian Huang
University of Michigan

Alexandre Gerber
AT&T Labs Research

Z. Morley Mao
University of Michigan

Subhabrata Sen
AT&T Labs Research

Oliver Spatscheck
AT&T Labs Research

ABSTRACT

Cellular networks employ a specific radio resource management policy distinguishing them from wired and Wi-Fi networks. A lack of awareness of this important mechanism potentially leads to resource-inefficient mobile applications. We perform the first network-wide, large-scale investigation of a particular type of application traffic pattern called *periodic transfers* where a handset periodically exchanges some data with a remote server every t seconds. Using packet traces containing 1.5 billion packets collected from a commercial cellular carrier, we found that periodic transfers are very prevalent in today's smartphone traffic. However, they are extremely resource-inefficient for both the network and end-user devices even though they predominantly generate very little traffic. This somewhat counter-intuitive behavior is a direct consequence of the adverse interaction between such periodic transfer patterns and the cellular network radio resource management policy. For example, for popular smartphone applications such as Facebook, periodic transfers account for only 1.7% of the overall traffic volume but contribute to 30% of the total handset radio energy consumption. We found periodic transfers are generated for various reasons such as keep-alive, polling, and user behavior measurements. We further investigate the potential of various traffic shaping and resource control algorithms. Depending on their traffic patterns, applications exhibit disparate responses to optimization strategies. Jointly using several strategies with moderate aggressiveness can eliminate almost all energy impact of periodic transfers for popular applications such as Facebook and Pandora.

Categories and Subject Descriptors

C.2.1 [**Computer-Communication Networks**]: Network Architecture and Design – Wireless Communication; C.4 [**Performance of Systems**]: Measurement Techniques

Keywords

Periodic transfers, Periodicity Detection, Smartphone Applications, Radio Resource Optimization, RRC State Machine, 3G Networks

1. INTRODUCTION

Cellular systems operate under restrictive constraints of resources including radio channel capacity, network processing capability, and handset energy consumption. A major U.S. carrier reported a growth of 5000% of its data traffic over 3 years [12], and all U.S. carriers are expected to spend $40.3 billion on cellular infrastructures in 2011 [1].

To efficiently utilize the limited resources and balance their incurred tradeoffs, cellular networks employ a resource management policy distinguishing them from wired and Wi-Fi networks. In particular, there exists a radio resource control (RRC) state machine [17] that determines radio resource usage based on application traffic patterns, affecting device energy consumption and user experience. A lack of understanding of such an important cellular resource control mechanism potentially leads to cellular-unfriendly (*i.e.,* resource-inefficient) mobile applications due to the poor interaction between their traffic pattern and the state machine. Equivalent resource control mechanisms with similar tradeoff considerations are deployed by different types of cellular networks such as UMTS [17], EvDO [11] and 4G LTE networks [20].

This paper presents the first network-wide, large-scale measurement of a popular type of mobile application traffic pattern called *periodic transfers* during which a handset (*i.e.,* a mobile device) periodically exchanges some data with a remote server. Our study is motivated by the following two key observations.

First, periodic transfers can be extremely resource-inefficient as they are small in size and short in duration relative to the periodicity. This is explained by the aforementioned cellular-specific resource management policy: unlike Wi-Fi and wired networks, in cellular networks, the release of radio resources (*i.e.,* demoting a handset from a high to a low-power state) is controlled by inactivity timers. The timeout value itself, also known as the *tail time* [10, 18], can last up to 17 seconds. Therefore, even for sending a tiny data burst containing one packet, a handset has to occupy the radio channel for at least 17 seconds due to the tail effect, thus wasting scarce radio resources and handset energy. Transferring such small bursts periodically (*e.g.,* every one minute) leads to even more serious resource inefficiencies.

Second, periodic transfers are typically *delay-tolerant* in that each periodic transfer instance is *not* initiated by a user. The typical way of generating them is to use a software timer with a fixed periodicity (*e.g.,* `java.util.Timer.scheduleAtFixedRate()` for Android). Therefore, there exists some leeway for smartphone applications to more intelligently reshape their traffic patterns to better match the characteristics of the network's radio resource management and thereby reduce their resource impact. Examples of such reshaping include adjusting the timing of each periodic transfer instance to overlap with user-triggered data transfers. Our analysis shows that by effectively applying traffic shaping techniques for periodic transfers, the network-wide resource impact of Facebook and Pandora can be reduced by up to 30% with very little impact on user experiences.

Cellular periodic transfers have been reported by pervious works [19,

WWW 2012, April 16–20, 2012, Lyon, France.
ACM 978-1-4503-1229-5/12/04.

15], which revealed that periodic transfers can be potentially extremely resource inefficient. In this paper, we perform a more in-depth and comprehensive study to quantitatively understand the following important characteristics:

- *Their **network-wide** prevalence in today's cellular traffic;*
- *Their impact on radio resources and handset battery life for commercial cellular networks;*
- *Their application-level semantics;*
- *Opportunities to make them more resource-efficient.*

None of the four aspects was addressed by previous work, and we investigate all of them in this measurement study. They are important to study because the findings will provide insights on how to fundamentally eliminate resource inefficiency caused by suboptimally scheduled traffic patterns, beyond periodic transfers. We detail our contributions as follows.

I. The first measurement of cellular periodic data transfer behavior in the wild (§4). Based on our measurement data consisting of 1.5 billion packets of UMTS (Universal Mobile Telecommunications System) traffic involving millions of users, we performed a detailed characterization of periodic transfers using a lightweight and robust periodicity detection algorithm. Our findings indicate that *(i)* periodic transfers are very prevalent in today's smartphone traffic as they occur in at least 20% of sessions longer than 1 minute within the dataset, *(ii)* periodic transfers are small and short with the median transfer size of 1.1KB and 90% of transfers shorter than 7 seconds, and *(iii)* the value of 60 seconds dominates the periodicity, which is likely to be set by developers in an ad-hoc manner. None of the above findings were previously known.

II. Exploration of the origins for periodic transfers (§5). This is a prerequisite for determining how to optimize such transfers without impacting any application semantic. By locally collecting and analyzing smartphone application traces, we found periodic transfers are caused by multiple factors such as polling, keep-alive messages for push-based services, advertisement transfers, and user-behavior measurement. In particular, we found popular applications that either unnecessarily (*e.g.,* Facebook) or aggressively (*e.g.,* Pandora and Textfree) issue periodic transfers, leading to significant waste of radio resources and handset energy.

III. Quantitative analysis of network-wide resource impact of periodic transfers (§6). Using the large dataset, we found that for popular applications such as Facebook and Pandora, periodic transfers account for less than 1.7% of traffic volume while their resource impact is as high as 30%. Even at the scope of *all* cellular data traffic for *all* applications, their radio energy impact (8%) is 20 times of their traffic volume contribution (0.4%). We have informed Pandora and Facebook of the problem caused by periodic transfers, and the responses were encouragingly positive [2].

IV. Detailed study of the effectiveness of optimization strategies on periodic transfers (§7). There exist various data scheduling, traffic shaping, and radio resource control algorithms for reducing resource consumption in cellular networks [9, 10, 15, 6, 18]. These strategies with critical tunable parameters incur complex yet important tradeoffs, which, although qualitatively known, are not quantitatively explored. To fully understand such tradeoffs, we quantify the effectiveness of various optimization strategies on periodic transfers (*e.g.,* scheduling them flexibly by overlapping them with non-periodic transfers). Our results indicate that jointly using multiple strategies with moderate aggressiveness can eliminate almost all energy impact (30%) of periodic transfers for Facebook and Pandora applications while incurring fewer side effects than aggressively employing one single strategy does. Such an approach greatly improves the battery life for end users and could lead to tremendous savings for cellular companies that invest billions of dollars each year on cellular network infrastructures [1].

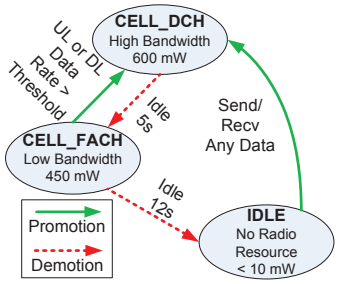

Figure 1: RRC State Machine of a large UMTS carrier in the U.S. The radio power consumption was measured by a Monsoon power monitor [3] on a Google Nexus One (HTC Passion) smartphone. More details can be found in [17].

Paper organization. §2 summarizes related work. §3 provides the background. We characterize cellular periodic transfers and their origins using a large dataset from a commercial UMTS carrier in §4 and §5. We identify the resource impact of periodic transfers in §6 and show the resource improvement brought by various optimization strategies in §7 before concluding the paper in §8.

2. RELATED WORK

Cellular periodic transfers were reported by the ARO (mobile Application Resource Optimizer) tool [19] that provides a platform for analyzing interactions between radio resource management and mobile applications. In [19], as a case study of a typical usage scenario, we used ARO to analyze *locally* collected Pandora traces to discover its periodic transfer behavior and resource inefficiency. We found from three Pandora traces that periodic data transfers only carry 0.2% of total bytes, but they account for 46% of total radio energy consumption and 40% of radio resource usage. Recent work by Kononen *et al.* [15] also identified the potential resource inefficiency of periodic transfers (without quantifying it) and proposed scheduling algorithms that reduce their energy consumption by up to 50%. Our study goes beyond previous works by systematically investigating the *network-wide* prevalence, application semantics, resource impact, and optimization of cellular periodic transfers.

Our study builds on previous work in measuring and optimizing cellular resource consumption. Previous work [17] characterizes the resource impact of the RRC state machine by analyzing a different dataset collected from a commercial UMTS network. Further, there exist various traffic shaping (*e.g.,* piggyback, batching, TailEnder [10], Intentional Networking [14], Time Alignment [15]) and radio resource control algorithms (*e.g.,* fast dormancy [6] and TOP [18]) for reducing resource consumption for cellular data transfers. We detail most of them in §7.1. In this study, we quantify the tradeoffs incurred by these techniques to maximize their efficacy on traffic patterns of periodic transfers.

3. BACKGROUND

Here we provide sufficient background, focusing on the popular 3G UMTS network where we obtained our measurement data.

3.1 The Cellular RRC State Machine

To efficiently utilize the limited radio resources, the UMTS radio resource control (RRC) protocol introduces a state machine associated with each handset. Typically there are three RRC states: IDLE (the default state when a handset is turned on, with no radio resource allocated), DCH (the high-power state enabling high-

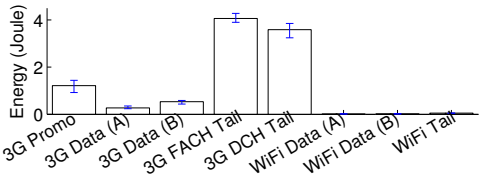

Figure 2: Radio energy breakdown for transmitting a small burst. A and B are two small HTTP objects of 1KB and 9KB, respectively.

speed data transmission), and FACH (the low-power state allowing only low-speed data transmission). Figure 1 shows the RRC state machine of one large commercial U.S. 3G UMTS carrier whose state transition model was inferred by previous work [17]. As illustrated, state *promotions* (going from a low-power to a high-power state) are triggered by user data transmission in either direction. State *demotions* (going in the reverse direction) are triggered by two inactivity timers configured by the RAN (Radio Access Network). At the DCH state, the RAN resets the DCH→FACH timer to a constant threshold T=5 seconds whenever it observes any data frame. If there is no user data transmission for T seconds, the DCH→FACH timer expires and the state is demoted to FACH. The FACH→IDLE timer uses a similar scheme.

Promotion Delays and Tail Times distinguish cellular networks from other types of access networks. An RRC state promotion incurs a long latency (up to 3 seconds) during which tens of control messages are exchanged between a handset and the RAN for resource allocation. A large number of state promotions increase signaling overhead at the RAN and worsen user experience [7]. In contrast, state demotions take negligible time, but they incur *tail times* that cause significant waste of resources [10, 18]. A *tail* is the idle time period matching the inactivity timer value before a state demotion. During the tail time, a handset still occupies transmission channels and WCDMA codes, and its radio power consumption is kept at the corresponding level of the state. Due to the tail time, transmitting very small amount of data can cause significant radio energy and radio resource consumption (§3.2).

Other Types of Radio Access Networks. Promotion delays and tail times also exist in other types of cellular RAN including GPRS/EDGE [5], EvDO [11], and 4G LTE networks [20]. For example, in 4G LTE networks, there are two RRC states: RRC_IDLE and RRC_CONNECTED (the low-power state is eliminated) [8]. We measured the inactivity timer from RRC_CONNECTED to RRC_IDLE to be 11.6s for a large LTE carrier in the U.S.

3.2 Small Data Transfer: 3G vs WiFi

We compare radio energy overhead of small data transfer for 3G and Wi-Fi to reveal the resource inefficiency of cellular periodic transfers, which consist of evenly spaced *small* data bursts. For 3G, we assume the handset is at IDLE state before transmitting a burst. Therefore the total radio energy consists of four parts: E_{Promo} (the IDLE→DCH promotion energy), $E_{\text{3G-Data}}$ (the energy for transferring the actual data), $E_{\text{DCH-Tail}}$ (the DCH tail energy), and $E_{\text{FACH-Tail}}$ (the FACH tail energy). For Wi-Fi, the radio energy consists of $E_{\text{WiFi-Data}}$ and $E_{\text{WiFi-Tail}}$, which are the energy for the data and the short tail, respectively.

We measured radio energy consumption for small data transfers by performing controlled local experiments. We set up an HTTP server hosting two small objects A and B, whose sizes are 1KB and 9KB, respectively (the median size of periodic transfers is 1.1KB as measured in §4.3). Then we used a Google Nexus One phone to fetch both objects for 20 times, making sure no caching takes place. Meanwhile, we recorded both power traces (using a Monsoon power monitor [3] with a sampling rate of 5,000 Hz) and packet traces (using tcpdump) whose timestamps were synchronized beforehand. We then correlated power traces with packet traces so that the energy consumption of each component (promotion, data and tail) could be accurately computed. All experiments were performed when the signal strength was good and stable. To determine the 3G radio power, we tried to keep other device components consuming constant power (*e.g.,* keeping the screen at the same brightness level). Then the 3G radio power, which contributes to 1/3 to 1/2 of the total handset power [19], can be approximated by subtracting the constant power baseline at the IDLE state (420 mW) from the overall handset power consumption reported by the power monitor.

Figure 2 plots the energy breakdown. Clearly the state promotion and tail time incur significant energy overhead for transmitting a small burst. For transferring Object A (B), 97.0% (94.3%) of radio energy belongs to E_{Promo}, $E_{\text{FACH-Tail}}$, or $E_{\text{DCH-Tail}}$. The Wi-Fi energy consumption is significantly less than that for 3G, because *(i)* Wi-Fi has smaller RTT and higher data rate than 3G, thus the data transfer time for Wi-Fi is much shorter, *(ii)* the Wi-Fi radio power (300 mW) is also smaller than 3G (650 mW), and *(iii)* Wi-Fi has a much shorter tail time (250 ms) and negligible state promotion delay. In our experiments, $E_{\text{3G-Data}}$ is 22 (31) times of $E_{\text{WiFi-Data}}$ for transferring Object A (B). When promotion and tail energy are taken into account, the disparity is even as high as 140 times.

Implications on periodic transfers. Periodic transfers are usually short in duration and small in size. This is not an issue for wired or Wi-Fi networks. For cellular networks, however, as demonstrated by our local experiment, *due to their promotion delays and tail times, periodic transfers may incur significant resource inefficiency.*

4. CHARACTERIZING CELLULAR PERIODIC TRANSFERS IN THE WILD

This section characterizes cellular periodic transfers using packet traces collected from a large UMTS carrier in the U.S.

4.1 3G Measurement Data

We analyzed a large packet header trace collected from a large 3G UMTS carrier in the U.S. for smartphones on Dec 29, 2010. The collection point is one GGSN (Gateway GPRS Support Node), and traffic from/to a subset of SGSNs (Serving GPRS Support Node) served by the GGSN are captured without any user, protocol, or flow-based sampling. The trace contains 1.5 billion packets continuously captured between 15:49 EST and 17:04 EST. We only recorded IP and transport-layer headers and a 64-bit timestamp for each packet, without any subscriber IDs or phone numbers, due to concerns of user privacy.

The major limitation of the dataset is its finite duration of 1.25 hours due to the storage limitation (1 Terabytes) although to our knowledge this is so far the cellular packet trace with the largest traffic volume. As recently reported by [23] using an aggregated one-week cellular dataset involving 600K subscribers and 22K smartphone apps, during daytime (9am to 5pm), traffic volume of most application categories remains stable (except for sports apps). We therefore expect our trace provides a largely representative snapshot of smartphone traffic patterns. Also note that the trace duration is much longer than the periodicities under investigation.

Subsequently, we extracted *sessions* from the trace, with each session consisting of all packets transferred by the same handset identified through the private client IP address present in the trace[1].

[1]Private IP addresses of our carrier are very stable. They change only at the interval of tens of minutes.

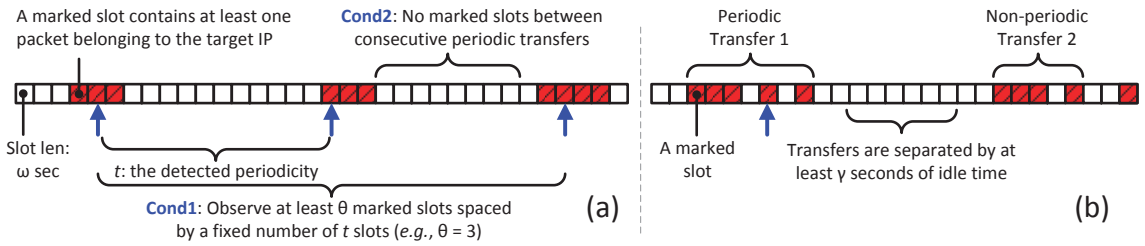

Figure 3: Periodicity detection algorithm (a) and identifying periodic transfers and their associated packets (b). Arrows are detected periodic seeds.

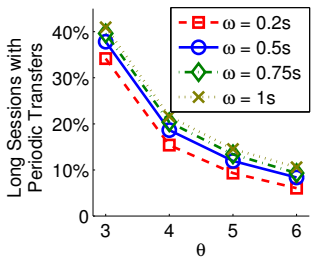

Figure 4: Percentage of long sessions (> 1min) with periodic transfers (change ω, θ).

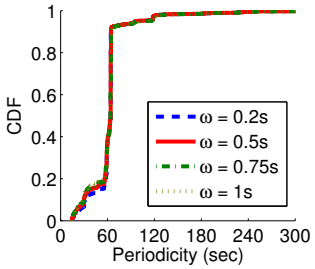

Figure 5: Distribution of detected periodicities ($\theta = 4$, change ω).

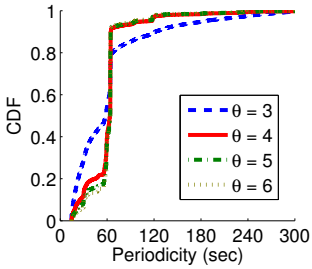

Figure 6: Distribution of detected periodicities ($\omega = 750$ms, change θ).

Multiple TCP or UDP flows may be mixed in one session. But we did not separate them as RRC state transitions are determined by the aggregated traffic of all applications on each handset. We used a threshold of 5 minutes of idle time to approximate the termination of a session. Changing this value to 3 or 6 minutes does not qualitatively affect the analysis results. We extracted about 2.8 million sessions from the dataset.

4.2 Detecting Periodicities

Performing periodic transfers with periodicity t means communicating with a particular server for every t seconds. Our periodicity detection algorithm takes a session (§4.1) as input, then focuses on each single server IP at a time to detect the periodicity of contacting that IP. For simplicity, we do not consider periodically contacting different server IPs (*e.g.,* contacting IP1 at t=0, IP2 at t=30s, IP3 at t=60s, *etc.*) as such cases are rare based on our observations of periodic DNS lookups (§5). But the algorithm does consider a case where a handset periodically contacts IP1 for a while, then switches to IP2. We also allow one session to have multiple periodicities. This happens when, for example, two TCP connections carrying periodic transfers coexist.

Why not use previous approaches? There are several existing techniques for such a common task of finding periodicities. Previous study [16] directly examines the frequency domain to extract the "TCP flow clock" (*i.e.,* regular spacing between flights of packets) by applying DFT on the packet time series then identifying peaks in the frequency spectrum. Previous work [21] employs auto-correlation to estimate RTT, which they assume causes equal spacing between bursts of packets. We found that neither method works well in our scenario because our interested periodicities (*e.g.,* 0.05Hz∼0.003Hz) have much fewer samples than RTTs (5Hz∼1Hz) do due to finite duration of sessions. ARO [19], which only analyzes short locally collected trace samples, uses a very simple approach by enumerating all $n(n-1)/2$ intervals in a time-series of n packets, incurring unacceptable complexity for our large dataset involving millions of long-lived sessions.

We instead propose a simple heuristic-based approach to effectively detect such "macroscopic" periodicities by *exhaustive search*. For each server IP i presented in the input session, we perform three steps to find the periodicity and periodic transfers associated with i (if exist). *(i)* Convert the continuous timestamps into discrete time slots to reduce noises and to speed up the search process. *(ii)* Search for repetitions of slots (containing packets of server i) spaced by a fixed timing gap t. Each of such detected slots is a "periodic seed" and t is the periodicity. *(iii)* Extract periodic transfers based on the periodic seeds. We now detail the three steps illustrated in Figure 3.

1. **Discretize timestamps** of packets of i into a binary array in which each slot has a fixed length of ω seconds. In other words, the k-th slot is marked if and only if there is at least one packet of i arrived within $[\omega k, \omega k + \omega)$[2]. Discretizing packet time series effectively tolerates noises caused by the network delay and reduces the computational overhead.

2. **Search for the periodicity.** The algorithm exhaustively searches for a periodicity of t slots, which is detected if and only if two conditions hold. First, we observe at least θ marked slots spaced by a fixed number of t slots *i.e.,* $\exists a$ such that slot a, $a + t$, ..., $a + (\theta - 1)t$ are all marked. Second, there is no marked slot between consecutive periodic transfers *i.e.,* $\forall 0 \le p < \theta, \forall 0 < b \le q \le t - b$: slot $a + pt + q$ should not be marked. b is a parameter tolerating the duration of a periodic transfer, which may occupy several slots from $a + pt - b$ to $a + pt + b$. We empirically choose $b = t/4$ but changing b to $t/5$ or $t/3$ has negligible impact on detection results. If both conditions hold, we mark slots a, $a+t$, ..., $a+(\theta-1)t$ as "periodic seeds" as indicated by arrows in Figure 3.

3. **Identify periodic transfers and their associated packets** for server i. A transfer consists of a succession of packets. In order to be periodic, a transfer must cover at least one periodic seed. Quantitatively, as shown in Figure 3b, we first cluster packets of i into *transfers* that are separated by at least γ seconds of idle time. Subsequently, a transfer of i is considered to be periodic if and only if at least one of its slots is a periodic seed.

The algorithm involves three key parameters: ω (the slot length), θ (the minimal repetitions to be observed before declaring a periodicity), and γ (the threshold for separating two transfers). Gen-

[2]We also mark slot $k - 1$ and $k + 1$ to prevent the algorithm from missing a periodicity due to the round-off error caused by discretization.

erally speaking, decreasing ω or increasing θ makes the algorithm more conservative and more confident in reporting periodicities. Increasing γ potentially makes identified (periodic) transfers larger in size and longer in duration. For example, we assume $\gamma = 2\omega$ in Figure 3b. Increasing γ to 3ω makes Transfer 2 further cover the rightmost slot. We investigate all above tradeoffs in §4.3.

Evaluation of the detection algorithm is challenging due to the lack of ground truth although the algorithm itself is intuitive. To validate our results, we built a program that visualizes the detected periodicities like Figure 3a. Then we manually inspected 100 randomly sampled sessions with at least one periodicity and 100 sessions with no periodicity detected (we used ω = 750ms and θ = 4 as justified in §4.3). The overall false negative rate is 3%. Here a "false negative" means a server IP in a session has obvious periodic transfer behavior identified by manual inspection but the algorithm missed it. One limitation of such a manual approach is that it is difficult to evaluate the false positive rate, which however is expected to be low due to discovered dominating periodicities (more than 90%) of 1 minute, 30 seconds, and 2 minutes (§4.3).

4.3 Measuring Periodicities

We apply the periodicity detection algorithm to the 2.8 million sessions of our dataset, and describe our findings.

How popular are periodic transfers? Figure 4 plots the percentage of sessions with at least one periodicity among all *long* sessions, whose durations are defined to be at least one minute, for different (ω, θ) pairs. Such long sessions account for only 35% of total sessions while they contribute to 98% of the traffic volume. Figure 4 indicates that the detection algorithm is more sensitive to θ than to ω. Given that 4 is a reasonable (yet still conservative) value of θ (justified later), we estimate periodic transfers occur in about 20% of long sessions. On the other hand, these 20% of long sessions contain almost 100% of detected periodicities.

Which periodicity values are commonly used? Figure 5 plots the CDF of detected periodicities for θ = 4 and four ω values. Figure 6 plots the same distribution for ω = 750ms and four θ values. The key observation is that a particular value of 60 seconds dominates the periodicities. Such a one-minute value is likely to be set by smartphone application developers in an ad-hoc manner. We also notice small clusters of 20 seconds, 30 seconds, and 2 minutes in Figure 5 and Figure 6. We learn from Figure 5 that the slot length ω has negligible impact on the periodicity distribution. However Figure 6 shows the distribution of θ = 3 differs from those of θ = 4, 5, 6 due to falsely identified small periodicities, most of which can be eliminated by increasing θ. Doing so however also reduces true positives. In particular, the algorithm will miss all periodic transfers with a periodicity of one minute, which dominates the periodicity distribution, occurring in sessions shorter than θ minutes. We pick θ = 4 which yields a reasonably good tradeoff between the accuracy and the coverage, as only about 10% of long sessions are between 3 and 4 minutes.

What are typical periodic transfer sizes? Figure 7 indicates that when γ, the idle timing gap threshold for separating consecutive transfers, is greater than 7 seconds, the 25th, 50th, and 75th percentiles of periodic transfer sizes remain very stable, with the values of 0.2KB, 1.1KB, and 1.8KB, respectively. Figure 8 confirms that distributions of periodic transfer sizes for γ = 5, 7.5, and 10sec are almost identical. Therefore, the extracted periodic transfers are insensitive to γ. We conclude that the vast majority of periodic transfers are small as 97% of periodic transfers are less than 10KB. Further, they are short relative to the periodicity in that 90% of them are shorter than 7 seconds (not shown in the Figures).

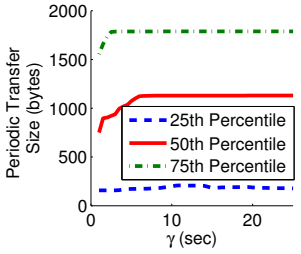

Figure 7: Relationship between γ and periodic transfer sizes.

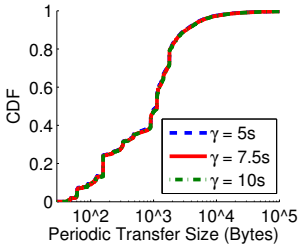

Figure 8: CDF of periodic transfer sizes for γ =5s, 7.5s, 10s.

Table 1: Applications generating periodic transfers. Periodicities listed below account for 46% of all detected periodic transfer instances.

Content Provider/ Applications	% Periodic Transfers	Remarks
facebook.com	48.4%	Keep connection alive for pushing
andomedia.com	15.5%	Pandora's audience measurement
medialytics.com	4.9%	User behavior monitoring
DNS	14.1%	DNS lookups
Advertisements	3.3%	Advertisement update
gmail.com	1.4%	Checking emails
pinger.com	1.4%	Polling to fetch updates for SMS
Other	11.1%	*e.g.,* checking weather

5. ORIGINS OF PERIODIC TRANSFERS

Understanding the origins of periodic transfers is a prerequisite for determining how to optimize such transfers without violating the application semantics. We do that in two steps.

First, we leverage a database containing IP address to content provider name mappings to understand which applications are responsible for periodic transfers. The database was generated by the carrier by examining the "Host" field of HTTP requests for the same set of SGSNs covered by the packet trace at a different time period (1.2 hours) on the same day when the packet trace was collected. Using the database, we found "meaningful" content providers, from which we can determine the application information, for IP addresses of 46% of detected periodicities (one session may contain multiple periodicities). For the remaining IP addresses, their content provider names (most are host names of CDN servers) are either useless for inferring the application information or not present in the database. Table 1 breaks down the origins of these 46% of periodicities.

Next, we locally collected and analyzed traces for the corresponding Android and iPhone applications to understand why those applications generate periodic transfers, whose intent can be largely classified into five categories, which are all *delay-tolerant* except for periodic DNS lookups.

1. Keep-alive Messages. Periodic transfers are used to prevent a TCP connection from being closed by the cellular NAT, whose timeout is much shorter than the TCP timeout (at least 2 hours by default [4]). We found when Facebook for iPhone starts, it sends an HTTP request to `facebook.com`. The server however does not send back the response immediately until either *(i)* a 60-second timeout is reached or *(ii)* it has a notification (*e.g.,* a message from a friend) to be pushed immediately to the client handset. In the first case, the server sends a keep-alive HTTP response (containing `for(;;);{"t":"continue"}`) to prevent a TCP connection from being shut down by the cellular NAT. When the handset receives the response, it immediately sends the next request and waits for the next response by following the same above procedure.

In its chatting mode, Facebook uses an even more aggressive periodicity of 20 seconds, which is slightly longer than the total

tail time (17 seconds shown in Figure 1), resulting in *(i)* a handset is almost always on the tail, and *(ii)* an IDLE→DCH promotion is triggered by each ping when no concurrent traffic exists, leading to very high signaling overhead.

Recent work [22] carried out local experiments to study NAT timeout behavior for four large cellular carriers in the U.S. Among them, three use a long timeout value of at least 20 minutes. The fourth carrier has the shortest timeout of 255 seconds before a TCP RST is sent to a handset. In any case, sending a keep-alive message every 60 seconds is too aggressive and incurs unnecessarily high resource overhead. We have shared our analysis with developers at Facebook and the reception has been enthusiastic.

2. Measurements. Mobile applications measure user behavior or preferences, and periodically send out collected information. For example, Pandora, the top mobile music streaming application, performs various audience measurements (*e.g.,* monitor online listeners' favorite radio stations) and uploads the measurement data to `andomedia.com` every one minute, which is too frequent as most songs are longer than one minute and Pandora is running in the background for most of the time. We have informed Pandora of the problem [2]. Their response was also positive and they agreed that the periodicity should be increased to at least 2 or 3 minutes based on their domain knowledge.

3. Polling. Periodic polling is observed in some applications. For example, Textfree, a popular SMS application, sends an HTTP request to `poll.pinger.com` every 20 seconds for querying for new short messages, and usually gets a "no-new-update" response. Due to its high energy overhead and delayed response, such a polling-based design is clearly worse than the push-based scheme used by, for example, Facebook.

4. Advertisements. Most, if not all, popular mobile advertisement platforms periodically refresh ads embedded in smartphone applications. For example, by default, AdMob uses a refresh rate of 60 seconds, while Mobclix aggressively updates the ad for every 15 seconds that is even shorter than the default tail time (17 sec, see Figure 1), making a handset persistently occupy the DCH or FACH state whenever the application containing an ad widget is running.

5. Periodic DNS lookups may happen before periodic transfers described above when persistent TCP connections are not used. Although handsets have local DNS caches, some content providers may set DNS TTLs to be small for load balancing (*e.g.,* 30 sec for Facebook). However, we observe that IP addresses returned by periodic DNS lookups seldom change for the same host name. This suggests that DNS-based load balancing is not frequently performed although content providers have such capabilities.

6. RESOURCE IMPACT

We use the large dataset described in §4.1 to study the resource impact of periodic transfers detected in §4.2 on the commercial UMTS network. Three metrics are used to quantify the resource impact. We leveraged the RRC state machine simulator and the handset power model used by the ARO tool [19] for computing the metrics below. The state machine simulator takes a packet trace as the input. It infers the RRC states experienced by the handset based on the timing, size, and direction of each packet, according to the state transition model shown in Figure 1.

- ***E***: the handset radio energy consumption. It is calculated by associating each RRC state or state promotion an average power value measured from a particular phone [17, 19][3].

[3]We measured power parameters of three handsets: Google Nexus One, HTC TyTn II, and HTC Dream. They yield similar ΔE values. Here we present the results using parameters of Nexus One (shown in Figure 1).

Table 2: Impact of periodic transfers at different scopes.

Study Scope	The contribution of periodic transfers. Traffic Volume	ΔE Radio Energy	ΔS Signaling Overhead	ΔD Radio Resources
U_0: All Sessions	0.4%	-7.9%	-8.9%	-6.5%
U_1: Periodic Sessions	0.7%	-20.4%	-25.3%	-15.6%
U_2:Facebook Sessions (μ=0.5)	1.7%	-30.5%	-30.4%	-30.5%
U_3:Pandora Sessions (μ=0.5)	0.5%	-28.7%	-35.0%	-20.5%

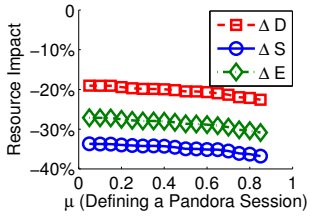

Figure 9: Impact of μ on Pandora sessions (U_3) in terms of ΔD, ΔS, and ΔE. A Pandora session is defined more strictly as μ increases.

- ***S***: the signaling overhead. It is quantified by the total state promotion delay.
- ***D***: the radio resource consumption. It is estimated by the total DCH occupation time (including the DCH tail time).

To investigate the impact of periodic transfers at different scopes, we study four session sets U_0 to U_3 listed in Table 2. U_0 is the entire dataset; U_1 refers to sessions that contain periodic transfers (they may also contain non-periodic transfers); U_2 and U_3 correspond to Facebook and Pandora sessions, respectively. It is trivial to identify sessions belonging to U_0 and U_1. A session (not necessarily containing periodic transfers) is considered to belong to U_2 if the fraction of bytes received from or sent to Facebook servers is at least μ. Pandora sessions (U_3) are defined in a similar way. We pick Facebook and Pandora since they are extremely popular [13] and both heavily use periodic transfers (§5).

For each study scope U_i, we quantify the resource impact of periodic transfers by *taking the difference* of computed metrics for the original trace and for the modified trace with all periodic transfers removed. Specifically, the radio energy impact of periodic transfers is computed as $\Delta E = (E_R - E_0)/E_0$ where E_0 and E_R correspond to the radio energy consumed by the original and the periodic-transfer-free trace, respectively, across all sessions of U_i. ΔE is negative as removing periodic transfers reduces energy consumption. The radio resource impact ΔD and the signaling impact ΔS are defined in similar ways.

The results are shown in Table 2. All results in §6 and §7 were generated using ω=750ms, θ=4 (defined in §4.2 and justified in §4.3), and μ=0.5. Clearly, there exists tremendous disparity between the traffic volume and the resource consumption of periodic transfers, indicating that *the state-of-art cellular periodic transfers are extremely resource inefficient*.

As an example, periodic transfers are responsible for only 0.5% of Pandora traffic (U_3), but their network-wide radio energy (ΔE), signaling overhead (ΔS), and radio resources (ΔD) impact are 40 to 70 times higher. Even at the scope of *all* cellular data traffic (U_0), their radio energy impact (8%) is 20 times of their traffic volume contribution (0.4%). Figure 9 shows how μ, controlling how U_3 is generated, affects ΔD, ΔS, and ΔE. For both Pandora and

Facebook, the resource impact of periodic transfers is considerable (at least 20%) even if μ is as low as 0.2.

7. OPTIMIZING PERIODIC TRANSFERS

Periodic transfers are prevalent but resource-inefficient. Fortunately, they have predictable periodicities and they are *delay-tolerant* in that the application usually has the flexibility over a time window in scheduling each transfer. There exist various data scheduling, traffic shaping, and radio resource control algorithms for reducing resource consumption for cellular data transfers (in particular, for delay-tolerant transfers). There however, remain several challenges in applying them effectively. *(i)* Depending on the diverse traffic patterns, various optimization strategies may result in different outcomes. Thus it is not trivial to select the right strategy for a specific application. *(ii)* The degree of aggressiveness of an optimization strategy, controlled by its critical parameters, often involves complex tradeoffs among handset radio energy, radio resources, signaling overhead, and user experience. Such tradeoffs, although qualitatively known, are not quantitatively and explicitly visible. *(iii)* Multiple strategies can be jointly applied. Their interaction can be even more complicated and is not well understood by previous work. To address these challenges, we make the aforementioned tradeoffs explicit by:

- **Evaluating the effectiveness** of traffic shaping and radio resource control techniques on periodic transfers.
- **Determining the level of aggressiveness** for resource optimization strategies by systematically exploring their parameter spaces.
- **Testing the compatibility** of multiple optimization techniques jointly applied to periodic transfers.

We next describe several representative optimization techniques in §7.1 before showing their optimization results in §7.2. Then in §7.3, we examine in detail the interplay between traffic patterns and various optimization techniques by performing case studies of three popular Android applications using locally collected traces.

7.1 Optimization Techniques

We consider four strategies (S_2 to S_5) below for optimizing periodic transfers. They can be applied individually or jointly to the input trace. We denote *target transfers* as our traffic of interest (*i.e.,* periodic transfers) to be optimized. The input trace contains both target and non-target transfers but the optimization strategies are applied to only target transfers (except for S_4). To our knowledge, S_2 to S_5 are representative and cover most *online* methods for optimizing (periodic) delay-tolerant transfers. They can be implemented in a generic library of a smartphone OS.

S_1**: The original trace without modification.** This corresponds to the comparison baseline.

S_2**: Piggybacking.** Target transfers can be shifted earlier, or be postponed till later, so that they can potentially be overlapped with non-target transfers, thus reducing the tail time. Clearly, user-triggered transfers cannot be shifted earlier. But periodic transfers can be transferred ahead of schedule since applications have complete control on when to initiate them. Assume that originally, a target transfer T occurs at time t_0, then after piggybacking, T can be transferred at any time between $t_0 - \delta$ and $t_0 + \delta$ where δ is a parameter defining an "elastic" window. If any non-target transfer happens within $(t_0 - \delta, t_0 + \delta)$, then T is piggy-backed with the *earliest* non-target transfer. Otherwise T is transferred at $t_0 + \delta$.

S_3**: Batching.** In order to reduce the overall tail time, several periodic transfer instances can be grouped into one single burst. Applying batching on periodic transfers essentially increases the periodicity[4]. It is performed by *(i)* creating batching points that are evenly spaced by k seconds from the first target transfer (k is a user-specified parameter determining the new periodicity), and *(ii)* scheduling each target transfer at its nearest batching point.

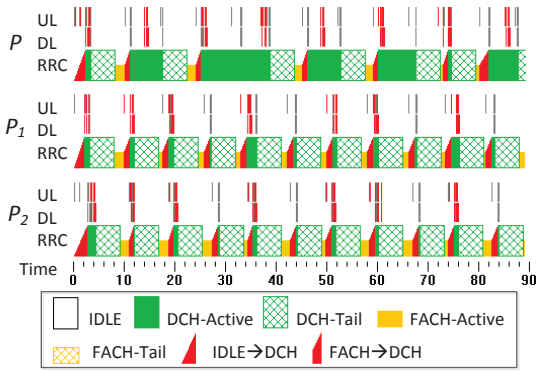

Figure 10: Validating the piggyback optimization S_2.

S_4**: Fast dormancy** is a recently proposed feature in 3GPP specifications [6, 17]. Instead of waiting for inactivity timers to expire, a handset can actively request for an *immediate* state demotion to IDLE by sending a special RRC control message to the radio access network. Fast dormancy can dramatically reduce the tail time while the potential penalty is increased signaling load (a positive ΔS) when it is aggressively used [7].

We consider the following algorithm for invoking fast dormancy: a handset maintains a *fast dormancy timer* (shorter than and independent to the RRC inactivity timers) that is reset whenever the handset sends or receives any packet. When the timer expires, the handset invokes fast dormancy to transition to IDLE regardless of the current RRC state or RRC timer value. Such behavior was observed on some baseband versions of Android smartphones. Fast dormancy does not change the traffic pattern. It can either be globally applied to all traffic, or be selectively used for only target transfers.

S_5**: TailEnder** [10] schedules transfers to minimize the energy consumption while meeting user-specified deadlines by delaying transfers and transmitting them together. It was implemented using the default parameters described in [10]. It is similar to piggyback while the major difference is that for TailEnder, a transfer can only be delayed but cannot be transferred ahead of schedule.

7.1.1 Implementation and Validation

We implemented offline versions of the four optimization strategies, allowing us to systematically exploring the optimization strategies and their parameter spaces using existing traces. Let P be the input trace and W be one or more optimization strategies. The *offline* optimizer computes P_1, the modified trace of P according to W. For validation purpose, we also implemented an *online* optimizer for S_2 and S_3 (§7.1) on a real smartphone (Samsung Galaxy S running Android 2.2). It initiates requests according to P. The requests are then scheduled and get transferred on the network based on the W, resulting in traffic pattern P_2. Ideally P_1 and P_2 should be the same.

We describe how we evaluate $W = \{S_2\}$. In this example shown in Figure 10, P contains two series of periodic transfers, C_1 and C_2, with the size of each transfer instance of 2 KB. The periodicities of C_1 and C_2 are 7 sec and 11 sec, respectively.

[4]For some applications of periodic transfers (*e.g.,* using them as keep-alive messages), batching may cause duplicate transfers, which are (conservatively) not removed due to the difficulty of identifying them.

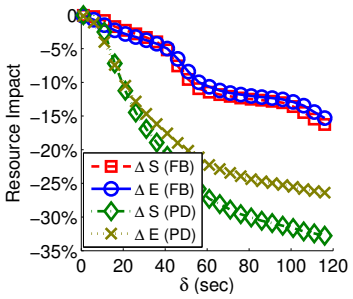

Figure 11: Impact of piggyback (S_2).

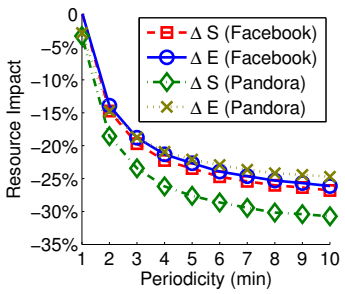

Figure 12: Impact of batching (S_3).

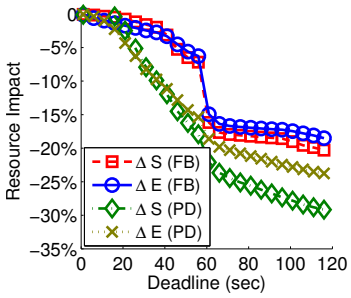

Figure 13: Impact of TailEnder (S_5).

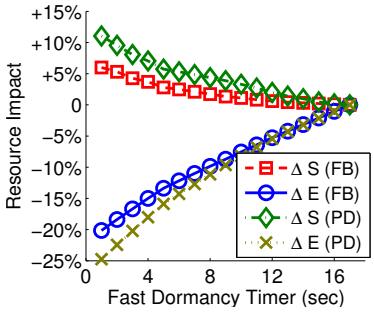

Figure 14: Impact of fast dormancy applied to only periodic transfers (S_4).

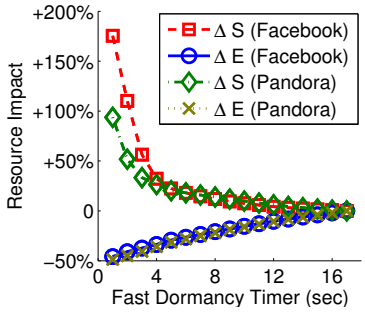

Figure 15: Impact of fast dormancy applied to all traffic (S'_4).

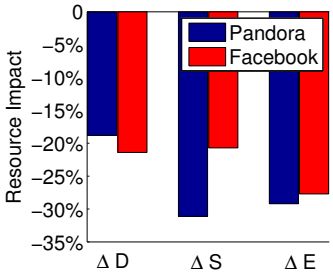

Figure 16: Resource savings of jointly using fast dormancy (the timer is 5s) for periodic transfers, piggyback (δ=30s), and batching (increasing the periodicity to 3 minutes).

$W = \{S_2\}$ corresponds to a scenario where transfers of C_2 (target transfers highlighted in red in Figure 10) are piggybacked with C_1 (non-target transfers) using an elastic window of δ=8 sec. Figure 10 shows that qualitatively P_1 and P_2 are very similar in terms of both traffic pattern and RRC states. Quantitatively, their total DCH time, FACH time, and promotion delay differ by no more than 3%. Changing the periodicities of C_1 and C_2 yields similarly small differences between P_1 and P_2, which are also similar for $W = \{S_3\}$ and $W = \{S_2, S_3\}$ (S_2 and S_3 are jointly applied) with varied parameters. The differences are mainly caused by network and promotion delays that vary between P_1 and P_2. The results indicate our offline optimizers are accurate.

7.2 Optimization Results

We consider the four types of optimizations described in §7.1: S_2 (piggyback), S_3 (batching), S_4 (fast dormancy applied to only periodic transfers), S'_4 (fast dormancy for all transfers), and S_5 (TailEnder [10]). The results are shown in Figures 11, 12, 14, 15, 13, respectively, for Facebook and Pandora. Similar to §6, we quantify the positive or negative impact of each optimization technique using ΔE and ΔS (ΔD is well correlated with ΔE) by comparing two scenarios where the optimization is disabled and enabled, respectively.

We highlight our findings as follows. *(i)* As expected, invoking batching, piggyback, and TailEnder more aggressively reduces more radio energy and signaling overhead at the cost of potentially degraded application functionality (*e.g.,* ad transfers may be delayed thus affecting real-time ad customization). The performance of piggyback and TailEnder are qualitatively similar. *(ii)* The side-effect of fast dormancy is increased signaling overhead as going to IDLE too quickly may incur an additional IDLE→DCH promotion triggered by the next packet. *(iii)* Blindly applying fast dormancy on *all* traffic is not recommended due to its prohibitively high signaling overhead increased by up to 175% (Figure 15). However, invoking fast dormancy only at the end of periodic transfers incurs acceptable signaling overhead while the achieved resource savings are still significant (Figure 14).

Further, we search combinations of multiple optimization techniques to reveal the merit of combining different strategies. Figure 16 considers a case where we jointly use batching (increasing the periodicity to 3 minutes, justified in §5 based on our contact with Pandora and our knowledge of cellular NAT), piggyback (δ = 30sec), and fast dormancy for periodic transfers (the timer is 5 seconds). Combining them essentially means increasing the periodicity to 3 minutes before piggybacking each periodic transfer with non-periodic transfers, then invoking fast dormancy at the end of each (shifted) periodic transfer if possible. Such a combined scheme eliminates almost all radio energy impact (about 30%) of periodic transfers for Facebook and Pandora.

To summarize our findings:

- All optimization strategies described in §7.1 effectively reduce the resource consumption of periodic transfers by incurring diverse tradeoffs.
- By jointly using several optimization strategies with moderate aggressiveness, the saved resources are comparable to those achieved by aggressively employing one single strategy with more significant side effects incurred.

7.3 Case Studies using Local Traces

We study three popular Android applications and show how to optimize their delay-tolerant transfers.

7.3.1 *Traces and Experimental Methodology*

Table 3 lists the three applications. *Pandora* is one of the most popular mobile music streaming applications. *Dictionary* corresponds to the top dictionary application on Android Market. *Pocket Express News* is a critically acclaimed news application. We refer to them as `Pandora`, `Dict`, and `News` henceforth. For each application, we analyze traces collected from five students who used the application as normal users. Clearly, the recorded traffic

Table 3: Case studies for three popular Android applications.

Trace	Application	# Traces	Length	Usage Scenario	Target Transfers
Pandora	Pandora Music	3	1.5∼2 hrs	Continuously playing the music	Periodic measurement (every 1 min)
Dict	Dictionary	5	10∼12 mins	Looking up an English word every 15 to 30 sec	Periodic ad (every 1 min)
News	Pocket Express News	5	14∼16 mins	Browsing and reading the news articles	Ad transfers triggered by user

Table 4: Effectiveness of optimization techniques for the three case study applications.

Resource impact relative to the overall resource consumption of the trace	Pandora			Dict			News		
	ΔE	ΔD	ΔS	ΔE	ΔD	ΔS	ΔE	ΔD	ΔS
Impact of target transfers	-47.7%	-39.2%	-57.3%	-12.6%	-23.9%	-9.8%	-42.3%	-48.3%	-42.9%
S_2: Piggyback ($\delta = 30$ s)	-19.3%	-16.7%	-18.8%	-6.5%	-8.6%	-6.6%	N/A		
S_3: Batching (every 3 mins)	-29.3%	-22.0%	-34.9%	-2.4%	+0.3%	+2.5%	N/A		
S_4: FD for target transfers (5s)	-24.1%	+2.1%	+8.9%	-0.2%	+9.4%	+12.3%	-18.4%	+5.9%	+11.4%
S_4': FD for all traffic (5s)	-34.4%	+3.1%	+12.5%	+1.5%	+62.3%	+64.8%	-28.7%	+11.3%	+17.1%
S_5: TailEnder (deadline 60s)	-19.7%	-15.4%	-20.5%	-3.1%	-3.5%	0.0	-19.1%	-19.3%	-23.5%

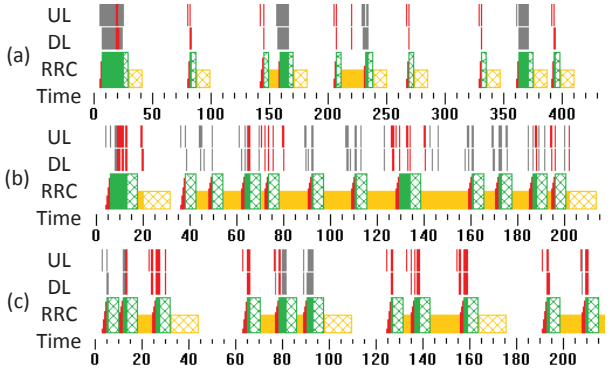

Figure 17: Traffic pattern examples of (a) Pandora, (b) Dict, (c) News. Target transfers are marked in red. Refer to Figure 10 for legends of the RRC states.

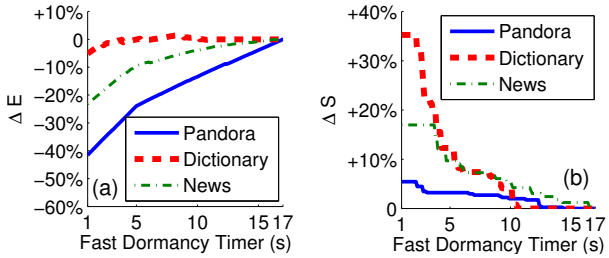

Figure 18: Impact of S_4 (fast dormancy for target transfers).

patterns are affected by user behavior randomness. We therefore focus on applications' most common usage scenarios (Table 3), which we found resulted in qualitatively similar traffic patterns among users. We are interested in usage patterns of the applications for a long period of usage time (at least several minutes) so that the optimization strategies for delay-tolerant transfers are effective.

7.3.2 *Traffic Patterns of the Three Applications*

Figure 17 plots traffic patterns and inferred RRC states for the three applications.

Traffic patterns of target transfers are highlighted in red in Figure 17. Both Pandora and Dict employ periodic transfers, which are automatically detected, for every one minute. News does not use periodic transfers. Instead, its advertisement transfers (manually labeled by us) are triggered when a user switches to a different news article or to the headline screen.

Traffic patterns of non-target traffic. As depicted in Figure 17, non-target traffic for Pandora is sparse and bursty, but each burst (music streaming) is transmitted using the maximal bandwidth. In contrast, packets for Dict are more evenly spread out, leading to a much higher ratio of radio channel occupation time to the total application usage time (86%). We found this is intrinsic to the application design of Dict, which makes delay-sensitive data inefficiently transferred by significantly under-utilizing the bandwidth, even if user requests are made at a moderate frequency of every 15 to 30 sec. For News, its ratio of radio channel occupation time (43%) lies between those of Pandora (21%) and Dict (86%).

7.3.3 *Optimization Results*

As summarized in Table 4, overall, the three applications exhibit different responses (in terms of resource savings) to optimization strategies due to their diverse traffic patterns. Note that piggyback and batching are not applicable to News, whose delay-tolerant transfers are initiated by a user.

The impact of target transfers. As indicated in Table 4, for Pandora and News, significant amount of radio energy (at least 42%) and signaling overhead (at least 43%) are spent on target transfers. For Dict, their contributions are only 12.6% for radio energy and 9.8% for signaling overhead, although Dict uses the same periodicity. This is because its non-target transfers already occupy radio channels for most of the time. Thus injecting target transfers only marginally increases resource utilization.

The impact of fast dormancy. S_4 and S_4' in Table 4 indicate that fast dormancy yields significantly different consequences on the three applications. For Pandora, fast dormancy effectively reduces resource consumption with very small signaling overhead (*i.e.,* state promotions) incurred, since the inter-burst time (illustrated in Figure 17a) is usually much longer than the fast dormancy timer value. For Dict, its traffic pattern consists of intermittent data bursts interleaved with short pauses, many of which are shorter than the fast dormancy timer. Therefore, invoking fast dormancy causes high signaling overhead. For example, by applying S_4', the total promotion delay increases by 40 sec for one Dict trace of 11 minutes, leading to even a positive value of ΔE. Such an application-dependent tradeoff is confirmed by Figure 18, which systematically investigates the impact of fast dormancy on ΔE (plot a) and on ΔS (plot b), for S_4. Aggressively invoking fast dormancy for target transfers benefits Pandora by bringing significant reduction of radio energy by up to 40% with signaling overhead of no more than 5%. On the other hand, fast dormancy is not applicable to Dict because the cost of achieving a saving of 5% of radio energy is the high signaling overhead of 35%.

The impact of piggyback and batching. For Pandora, S_2 and S_3 in Table 4 indicate that both piggyback and batching effectively save resources (TailEnder is qualitatively similar to piggyback). For Dict, the resource savings brought by piggyback and batching

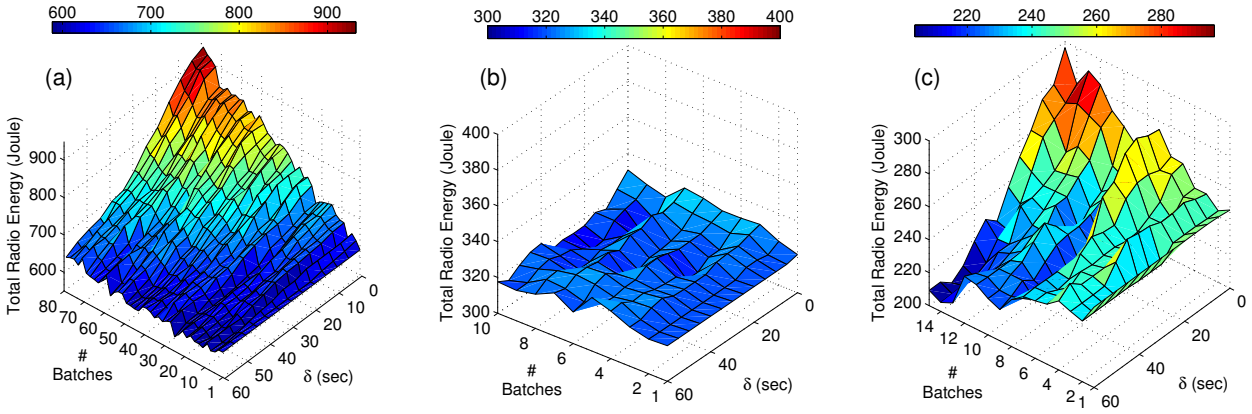

Figure 19: Jointly applying piggyback (varying δ), batching (varying the number of batches), and fast dormancy (the timer is 5s) on target transfers for the three apps: (a) `Pandora` (b) `Dict` (c) `News`. Piggyback and batching can be (a) compatible, (b) ineffective, or (c) incompatible.

are very limited due to its inefficient non-target transfers. Figure 19 shows how each application is jointly optimized by three strategies: piggyback, batching, and fast dormancy (only for target transfers), revealing interesting interactions between piggyback and batching. For `Pandora`, either increasing δ or decreasing the number of batches (*i.e.,* increasing the periodicity) effectively reduces radio energy consumption. However, neither strategy is effective for `Dict` due to its non-target transfers that are already spread out. Since piggyback and batching are not applicable to `News`, we just use its traffic pattern to demonstrate a potential type of interaction between piggyback and batching. As shown in Figure 19c, jointly applying piggyback and batching does not outperform the piggyback-only approach, which effectively overlaps almost all target transfers with non-target transfers. In that case, aggressively using batching actually *wastes* radio energy since one long batched transfer is not as flexible as several short transfers when being overlapped with non-target transfers (unlike `Pandora` whose batched transfers are still short enough).

To summarize, our case study indicates that it is far from trivial to determine what combination of techniques and for each technique what parameter settings to use for balancing the tradeoff. Effective solutions require efforts from both application and platform developers. We summarize our recommendations as follows.

- For `Pandora`, fast dormancy, batching, and piggyback can be jointly and aggressively used whenever timing constraints of audience measurement are satisfied. The savings of radio energy and signaling overhead are 40% when the parameters of S_2, S_3, and S_4 are 60s, every 3mins, and 5s, respectively (similar to our findings of Pandora in §7.2).
- For `Dict`, the inefficient non-target transfers need to be improved, otherwise the optimization is largely ineffective.
- For `News`, TailEnder (a deadline of 60 sec) and fast dormancy (5s for target transfers) can be applied to reduce radio energy and signaling overhead by up to 25%.

8. CONCLUDING REMARKS

Using large packet traces collected from a commercial UMTS network, we performed the first network-wide investigation of cellular periodic data transfers, which have significant impact on resource consumption despite contributing only small traffic volumes. Periodic transfers are widespread and are generated for various reasons, many of which are overly aggressive, if not unnecessary. We also investigated how well various traffic shaping and resource control algorithms might be used to optimize resource utilization for periodic transfers. Our work is an important step towards better understanding cellular traffic dynamics and the complex interactions between radio resource control policy and mobile application behavior, and determining how to develop cellular-friendly mobile applications.

9. REFERENCES

[1] `http://pennysleuth.com/invest-in-cell-phone-infrastructure-for-growth-in-2010/`.
[2] A Call for More Energy-Efficient Apps. `http://www.research.att.com/articles/featured_stories/2011_03/201102_Energy_efficient`.
[3] Monsoon Power Monitor. `http://www.msoon.com/`.
[4] TCP Man page. `http://linux.die.net/man/7/tcp`.
[5] GERAN RRC State Mchine. 3GPP GAHW-000027, 2000.
[6] UE "Fast Dormancy" behavior. 3GPP discussion and decision notes R2-075251, 2007.
[7] System Impact of Poor Proprietary Fast Dormancy. 3GPP discussion and decision notes RP-090941, 2009.
[8] 3GPP TS 36.331: Radio Resource Control (RRC) (V10.3.0), 2011.
[9] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and Transparent Dynamic Content Updates for Mobile Clients. In *Mobisys*, 2006.
[10] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications. In *IMC*, 2009.
[11] M. Chatterjee and S. K. Das. Optimal MAC State Switching for CDMA2000 Networks. In *INFOCOM*, 2002.
[12] J. Erman, A. Gerber, M. Hajiaghayi, D. Pei, S. Sen, and O. Spatscheck. To Cache or not to Cache: The 3G case. *IEEE Internet Computing*, 2011.
[13] A. Gember, A. Anand, and A. Akella. A Comparative Study of Handheld and Non-Handheld Traffic in Campus Wi-Fi Networks. In *PAM*, 2011.
[14] B. Higgins, A. Reda, T. Alperovich, J. Flinn, T. Giuli, B. Noble, and D. Watson. Intentional Networking: Opportunistic Exploitation of Mobile Network Diversity. In *Mobicom*, 2010.
[15] V. Kononen and P. Paakkonen. Optimizing Power Consumption of Always-On Applications Based on Timer Alignment. In *COMSNETS*, 2011.
[16] F. Qian, A. Gerber, Z. M. Mao, S. Sen, O. Spatscheck, and W. Willinger. TCP Revisited: A Fresh Look at TCP in the Wild. In *IMC*, 2009.
[17] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. Characterizing Radio Resource Allocation for 3G Networks. In *IMC*, 2010.
[18] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck. TOP: Tail Optimization Protocol for Cellular Radio Resource Allocation. In *ICNP*, 2010.
[19] 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.
[20] S. Sesia, I. Toufik, and M. Baker. LTE: The UMTS Long Term Evolution From Theory to Practice. John Wiley and Sons, Inc., 2009.
[21] B. Veal, K. Li, and D. Lowenthal. New Methods for Passive Estimation of TCP Round-Trip Times. In *PAM*, 2005.
[22] Z. Wang, Z. Qian, Q. Xu, Z. M. Mao, and M. Zhang. An Untold Story of Middleboxes in Cellular Networks. In *SIGCOMM*, 2011.
[23] Q. Xu, J. Erman, A. Gerber, Z. M. Mao, J. Pang, and S. Venkataraman. Identifying Diverse Usage Behaviors of Smartphone Apps. In *IMC*, 2011.