Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications

Niranjan Balasubramanian Aruna Balasubramanian Arun Venkataramani
Department of Computer Science
University of Massachusetts Amherst
{niranjan, arunab, arun}@cs.umass.edu

ABSTRACT

In this paper, we present a measurement study of the energy consumption characteristics of three widespread mobile networking technologies: 3G, GSM, and WiFi. We find that 3G and GSM incur a high *tail energy* overhead because of lingering in high power states after completing a transfer. Based on these measurements, we develop a model for the energy consumed by network activity for each technology.

Using this model, we develop TailEnder, a protocol that reduces energy consumption of common mobile applications. For applications that can tolerate a small delay such as e-mail, TailEnder schedules transfers so as to minimize the cumulative energy consumed while meeting user-specified deadlines. We show that the TailEnder scheduling algorithm is within a factor $2\times$ of the optimal and show that any online algorithm can at best be within a factor $1.62\times$ of the optimal. For applications like web search that can benefit from prefetching, TailEnder aggressively prefetches several times more data and improves user-specified response times while consuming *less* energy. We evaluate the benefits of TailEnder for three different case study applications—email, news feeds, and web search—based on real user logs and show significant reduction in energy consumption in each case. Experiments conducted on the mobile phone show that TailEnder can download 60% more news feed updates and download search results for more than 50% of web queries, compared to using the default policy.

Categories and Subject Descriptors

C.2 [**Computer Communication Network**]: Network Architecture and Design— *Wireless Communication*

General Terms

Algorithms, Design, Performance

Keywords

Cellular networks, WiFi, Power measurement, Energy savings, Mobile applications

IMC'09, November 4–6, 2009, Chicago, Illinois, USA.

1. INTRODUCTION

Mobile phones are ubiquitous today with an estimated cellular subscription of over 4 billion worldwide [2]. Most phones today support one or more of 3G, GSM, and WiFi for data transfer. For example, the penetration of 3G is estimated at over 15% of cellular subscriptions worldwide and is over 70% in some countries [1].

How do the energy consumption characteristics of network activity over 3G, GSM, and WiFi on mobile phones compare with each other? How can we reduce the energy consumed by common applications using each of these three technologies? To investigate these questions, we first conduct a detailed measurement study to quantify the energy consumed by data transfers across 3G, GSM, and WiFi. We find that the energy consumption is intimately related to the characteristics of the workload and not just the total transfer size, e.g., a few hundred bytes transferred intermittently on 3G can consume more energy than transferring a megabyte in one shot. Below is a summary of the key findings of our measurement study, which remain consistent across three different cities, diurnal variation, mobility patterns, and devices.

1. In 3G, a large fraction (nearly 60%) of the energy, referred to as the *tail energy*, is wasted in high-power states after the completion of a typical transfer. In comparison, the *ramp energy* spent in switching to this high-power state before the transfer is small. Tail and ramp energies are constants that amortize over larger transfer sizes or frequent successive transfers.

2. In GSM, although a similar trend exists, the time spent in the high-power state after the transfer, or the *tail time*, is much smaller compared to 3G (6 vs. 12 secs). Furthermore, the lower data rate of GSM implies that more energy is spent in the actual transfer of data.

3. In WiFi, the association overhead is comparable to the tail energy of 3G, but the data transfer itself is significantly more efficient than 3G for all transfer sizes.

Based on these findings, we develop a simple model of energy consumption of network activity for each of the three technologies. We utilize these models to identify opportunities for reducing the energy consumption of network activity induced by common mobile applications. To this end, we design TailEnder, an energy-efficient protocol for scheduling data transfers. TailEnder considers two classes of applications: 1) delay-tolerant applications such as email and RSS feeds, and 2) applications such as web search and web browsing that can benefit from aggressive prefetching.

For delay-tolerant applications on 3G and GSM, TailEnder schedules outgoing transfers so as to minimize the overall time spent in

high energy states after completing transfers, while respecting user-specified delay-tolerance deadlines. We show that the TailEnder scheduling algorithm is provably within a factor $2\times$ of the energy consumed by an optimal offline algorithm that knows the complete arrival pattern of transfers a priori. Furthermore, we show that no deterministic online algorithm can be better than 1.62-competitive with respect to an optimal offline adversary.

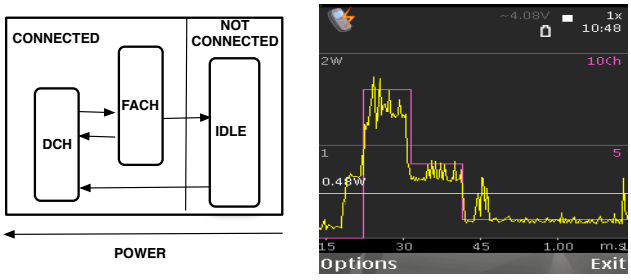
For applications that can benefit from prefetching, TailEnder determines what data to prefetch so as to minimize the overall energy consumed. Prefetching useful data reduces the number of transfers and their associated cumulative tail energy, while prefetching useless data incurs additional transmission energy. TailEnder uses a probabilistic strategy to balance these concerns. Somewhat counterintuitively, for applications such as web search, TailEnder fetches several times more data and improves user-perceived response times, but still consumes *less* energy.

We evaluate the performance of TailEnder for three different applications: email, news feeds and web search. For each of these applications, we collect real user traces including arrival times and transfer sizes. We evaluate TailEnder by conducting experiments on the mobile phone and find that TailEnder can download 60% more news feed updates and download search results for more than 50% of web queries, compared to using the default policy. Our model-driven simulation shows that TailEnder can reduce energy by 35% for email applications, 52% for news feeds and 40% for web search. Further, we find that, opportunistic WiFi access substantially reduces energy consumption compared to only using 3G. Even when WiFi in only available 50% of the time, sending data over WiFi when available reduces the energy consumption by over 3 times for all three applications.

2. BACKGROUND AND RELATED WORK

2.1 Cellular power management

Two factors determine the energy consumption due to network activity in a cellular device. First, is the transmission energy that is proportional to the length of a transmission and the transmit power level. Second, is the *Radio Resource Control* (RRC) protocol that is responsible for channel allocation and scaling the power consumed by the radio based on inactivity timers.



(a) (b)

Figure 1: (a) The radio resource control state machine for 3GPP networks consisting of three states: IDLE, DCH and FACH (b) Instantaneous power measurements for an example transfer over 3G showing the transition time between high to low power state

Figure 1(a) shows the state machine [18] implemented by the RRC protocol for GSM/EDGE/GPRS (2.5G) as well as UMTS/WCDMA (3G) networks that follow the 3GPP [5] standard. The radio remains in the IDLE state in the absence of any network activity. The radio transitions to the higher power states, DCH (Dedicated Channel) or FACH (Forward Access Channel), when the network is active. The DCH state reserves a dedicated channel to the device and ensures high throughput and low delay for transmissions, but at the cost of high power consumption. The FACH state shares the channel with other devices and is used when there is little traffic to transmit and consumes about half of the power in the DCH state. The IDLE state consumes about one percent of the power in the DCH state.

The transition between the different states is controlled by inactivity timers [18]. Figure 1(b) shows the instantaneous power measurements for an example transfer. The graph shows the time taken to transition from a high power to a low power state. Instead of transitioning from the high to the low power state immediately after a packet is transmitted, the device transitions only when the network has been inactive for the length of the inactivity timer. This mechanism serves two benefits: 1) it alleviates the delay incurred in moving to the high power state from the idle state, and 2) it reduces the signaling overhead incurred due to channel allocation and release during state transitions. Since lingering in the high power state also consumes more energy, network operators set the value of the inactivity timer based on this peformance/energy trade-off [18, 12], with typical values being several seconds long.

The 3GPP2 standard [4] used by the CDMA2000 technology is another standard for 3G networks, and 3GPP and 3GPP2 are the most prevalent standards today. Though the state machine for the radio resource control in the 3GPP2 standard is different from that shown in Figure 1, several features are similar. In particular, the 3GPP2 standard also uses an inactivity timer to transition from the high power to the low power state for performance reasons [18, 25].

2.2 WiFi power management

In comparison, WiFi incurs a high initial cost of associating with an access point (AP). However, because WiFi on phones typically uses the Power Save Mode (PSM), the cost of maintaining the association is small. When associated, the energy consumed by a data transfer is proportional to the size of the data transfer and the transmit power level. Our measurements (Section 3) confirm that the transmission energy consumed by WiFi is significantly smaller than both 3G and GSM, especially for large transfer sizes.

2.3 Related work

Energy consumption of network activity in mobile phones has seen a large body of work in recent times. To our knowledge, our paper presents the first comparative study of energy consumption characteristics of all three technologies—3G, GSM, and WiFi—that are under widespread use today. In particular, our study of the energy consumption characteristics of 3G reveals significant and nonintuitive implications for energy-efficient application design.

Analytical modeling. Prior work [18, 25, 12] has studied the impact of different energy saving techniques in 3G networks using analytical models. These works analyze the impact of the inactivity timer by modeling the delay and energy utilization for different values for the inactivity timer. Their goal is to determine the optimal value of the inactivity timer from the network operator's perspective. Yeh *et al.* [25] analytically compare the impact of the inactivity timer in both 3GPP and 3GPP2 networks. In comparison, our study treats the inactivity timer value as a given and develops algorithms for energy-efficient application design based on real measurements and application traces.

Measurement. Gupta *et al.* [14] present a measurement study

of the energy consumption of VoIP applications over WiFi-based mobile phones. The authors find that intelligent scanning strategies and aggressive use of PSM in WiFi can reduce power consumption for VoIP applications. Xiao *et al.* [24] measure the energy consumption for Youtube-like video streaming applications in mobile phones using both WiFi and 3G. Their focus is on the energy utilization of various storage strategies and application-level strategies such as delayed-playback and playback after download. Nurminen *et al.* [20] measure the energy consumption for peer-to-peer applications over 3G. In comparison to these application-specific studies, our focus is on reducing the energy consumption of general network activity across 3G, GSM, and WiFi.

Energy-efficient mobile network activity. Several previous studies [6, 22, 23, 7] have investigated strategies for energy-efficient network activity in mobile phones supporting multiple wireless technologies. Pering et al. [22] develop strategies to intelligently switch between WiFi and Bluetooth. Agarwal et al. [6] propose an architecture to use the GSM radio to wake up the WiFi radio upon an incoming VoIP call to leverage the better quality and energy-efficiency of WiFi while keeping its scanning costs low. Krashinsky *et al.* [17] present an alternative to the 802.11 power saving mode to minimize energy consumption over WiFi. The authors present an adaptive algorithm that minimizes energy by determining the length of time that the WiFi interface should be switched off based on network activity without affecting performance.

Rahmati *et al.* [23] show that intelligently switching between WiFi and GSM reduces energy consumption substantially as WiFi consumes less transmission power. However, in order to avoid the cost of unnecessary scanning in the face of poor WiFi availability, the authors design an algorithm that predicts WiFi availability, and the device scans for WiFi access points only in areas where WiFi is available with high probability. Trevor *et al.* [7] present application-level modifications to reduce energy consumption for updates to dynamic web content. Their key ideas include using a proxy to 1) only push new content when the portion of the web document of interest to the user is updated, 2) batch updates to avoid the overhead of repeated polling, and 3) use SMS on GSM to signal the selection of WiFi or GSM based on the transfer size for energy-efficient data transfer. In addition to confirming these prior findings about GSM and WiFi power consumption, our measurement study also investigates 3G that reveals significantly different energy consumption characteristics, which lead us to develop novel energy-efficient data transfer algorithms for 3G.

Algorithms. Prior theoretical works [11, 13, 15, 8] study the problem of energy minimization while meeting job deadlines in processors that transition between the sleep and active state. The model assumed in most of these works is that a device incurs a large *ramp energy* overhead in transitioning from the low power state to the high power state, so the goal is to minimize the number of transitions. As explained in Sections 3 and 4, this model cannot be applied *as is* to mobile phones because their transition characteristics are different and include a significant *tail energy* component.

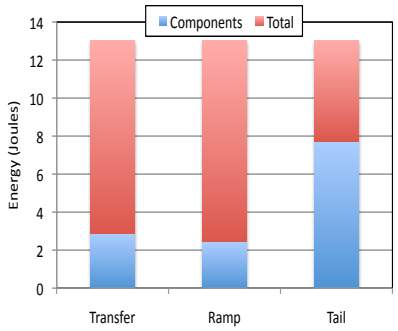
3. MEASUREMENT

We conduct a measurement study with the following goals:

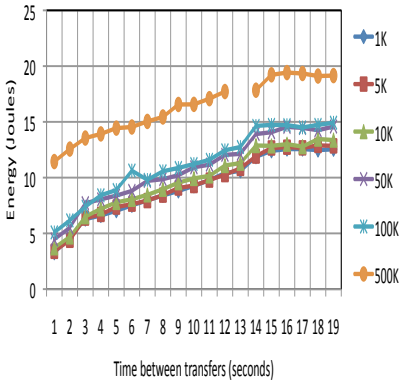
1. Compare the energy consumption characteristics of 3G, GSM and WiFi and measure the fraction of energy consumed for data transfer versus overhead.

2. Analyze the variation of the energy overhead with geographic location, time-of-day, mobility, and devices.

3. Develop a simple energy model to quantify the energy consumption over 3G, WiFi and GSM as a function of the transfer size and the inter-transfer times.



(a) 3G: Energy components



(b) 3G: Varying inter-transfer times

Figure 2: 3G Measurements: (a) Average ramp, transfer and tail energy consumed to download 50K data. The lower portion of the stacked columns show the proportion of energy spent for each activity compared to the total energy spent. (b) Average energy consumed for downloading data of different sizes against the inter-transfer time

3.1 Devices and Tools

The majority of our experiments are performed using four Nokia N95 phones (http://en.wikipedia.org/wiki/Nokia_N95). Two of the phones are 3G-enabled AT&T phones that use HSDPA/UMTS technology and two are GSM-enabled AT&T phones that use EDGE. All four phones were equipped with an 802.11b WiFi interface. We use Python, PyS60 v1.4.2, developed for the Symbian OS 3rdEd FP 1 to conduct data transfer experiments. To measure energy consumption, we use Nokia's energy profiling application, the Nokia Energy Profiler (NEP) v1.1 [1]. NEP provides instantaneous power measurements sampled once every 250 milli-seconds. Using the power measurements, we estimate the energy consumed by approximating the area under the power measurement curve over a time interval. Unless stated, all measurement results are averaged over 20 trials and error bars show the 95% confidence interval.

We also report results from a smaller scale measurement study performed on the HTC Fuze phone that runs Windows Mobile 6.5. The HTC phone is also a 3G-enabled AT&T phone and is equipped

[1]http://www.forum.nokia.com/Resources_and_Information/Tools/Plug-ins/Enablers/Nokia_Energy_Profiler).

with an 802.11b WiFi interface.The energy measurements on the HTC phone were performed using a hardware power meter [3] that measured power once every 0.2 milliseconds.

3.2 Measurement Methodology

3.2.1 3G and GSM

Our 3G and GSM measurements quantify the: 1) *Ramp energy*: energy required to switch to the high-power state, 2) *Transmission energy*, and 3) *Tail energy*: energy spent in high-power state after the completion of the transfer.

We conduct measurements for data transfers of different sizes (1 to 1000 KB) with varying intervals (1 to 20 seconds) between successive transfers. We measure energy consumption by running NEP in the background while making data transfers. For each configuration of (x, t), where $x \in$ [1K, 1000K] and $t \in$ [1, 20] seconds, the data transfers proceed as follows: The phone initiates an x KB upload/download by issuing a http-request to a remote server. After the upload/download is completed, the phone waits for t seconds and then issues the next http request. This process is repeated 20 times for each data size. Between data transfer experiments for different intervals, the phone remains idle for 60 seconds. The energy spent during this period is subtracted from the measurements as idle energy. We extract the energy measurements from the profiler for analysis, and use the time-stamps recorded by NEP to mark the beginning and end of data transfer as well as the beginning and end of the *Ramp time* and the *tail-time*. The energy consumed by each data transfer is computed as the area under the power-curve between the end of *Ramp time* and the start of *tail-time*.

3.2.2 WiFi

Our WiFi measurements quantify the energy : 1) to scan and associate to an access point and 2) to transfer data. We conduct two sets of measurements. In the first set of measurements, for each data transfer, we first scan for WiFi access points, associate with an available AP and then make the transfer. In the second set of measurements, we only make one scan and association for the entire set of data transfers to isolate the transfer energies.

In addition, all three networks, 3G, GSM and WiFi, incur a maintenance energy, which is the energy used to keep the interface up. We estimate the maintenance energy per second by measuring the total energy consumed to keep the interface up for a time period.

3.2.3 Accounting for idle power

For all measurements, we configure the phone in the lowest power mode and turn off the display and all unused network interfaces. The energy profiler itself consumes a small amount of energy, which we include in the idle power measurement. We measure idle energy by letting the energy profiler run in the background with no other application activity. The average idle power is less than 0.05 W and running the energy profiler at a sampling frequency of 0.25 seconds increases the power to 0.1 W.

3.3 3G Measurements

Figure 2(a) shows the average energy consumption for a typical 50KB download over 3G. We find that the *Tail energy* is more than 60% of the total energy. The *Ramp energy* is significantly small compared to the tail energy, and is only 14% of the total energy. 3G also incurs a maintenance energy to keep the interface on, and is between 1-2 Joules/minute (not shown).

Figure 2(b) shows the average energy consumed for download when the time between successive transfers is varied. We ignore the idle energy consumed when waiting to download the next packet. Consider the data points for downloading 100 KB data. The energy increases from 5 Joules to 13 Joules as the time between successive downloads increases from 1 second to 12 seconds. When the time between successive downloads is greater than 12.5 seconds, the energy consumed for 100 KB transfers plateaus at 15 Joules. When the device waits less than the *tail-time* to send the next packet, each data transfer does not incur the total *Tail energy* penalty, reducing the average energy per transfer. This observation suggests that the *Tail energy* can be amortized using multiple transfers, but only if the transfers occur within *tail-time* of each other. This observation is crucial to the design of TailEnder, a protocol that reduces the energy consumed by network applications running on mobile phones.

3.3.1 Geographical and Temporal variations

We measure the energy consumption across different days and in different geographical regions. The objectives of the experiment are 1) to verify that mobile phones in different cell tower areas are affected by the *tail-time* overhead, and 2) to measure the temporal consistency of *tail-time*.

Figure 3(a) shows that the *Tail energy* remains consistent across three days. On the other hand, Figure 3(b) shows that the *Ramp energy* is about 2 and 4 Joules for the measurements conducted on different days.

We conducted 3G energy measurements in three different cities, Amherst, Northamption and Boston, in Massachusetts, USA using two different devices. Figure 3(c) shows that the *Tail energy* is consistent across different locations and two different Nokia devices (D1 and D2). The figure also shows that the *Tail energy* and *Ramp energy* do not vary across day (9:00 am to 5:00 pm) and night (8:00 pm to 6:00 am). The measurement results provide additional evidence that the *tail-time* or the inactivity timer is configured statically by network operators and can be inferred empirically. In Section 4, we use the value of the inactivity timer to design TailEnder.

Figure 4(a) compares the average energy consumed by downloads during the day versus night, averaged over 3 days of transfer data. Although the ramp and tail energies are similar during night and day (shown in Figure 3(c)), the energy consumed during the night is up to 10% lower than during the day. This is likely due to lower congestion during the night.

3.3.2 Uploads

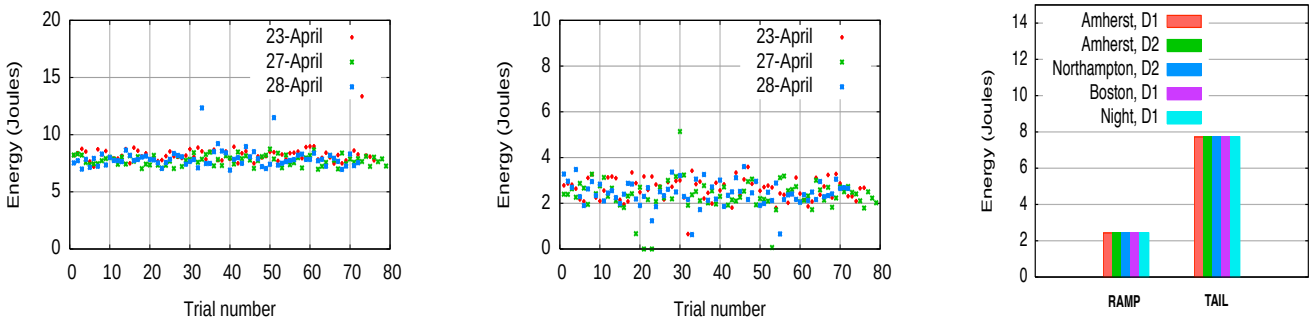
Figure 4(b) shows the average tail, ramp and transfer energy for upload experiments. As observed in the download experiments, the *Tail energy* consumes more than 55% of the total energy. Figure 4(c) shows that the transfer energy for uploads is higher than downloads for larger data sizes. For example, the transfer energy for uploads is nearly 30% more than that for downloads for 100 KB transfers. One cause for this difference is that upload bandwidths are typically smaller than the download bandwidth.

3.3.3 Mobility

Figure 5 compares energy consumption under mobility within the town of Amherst, MA for 50K data transfers. Mobility in outdoor settings affect transfer rates due to factors such as signal strength and hand-offs between cell towers, resulting in varying transfer times [19]. Despite the large variances in the transfer energy compared to the stationary measurements, we observe that the *Tail energy* accounts for nearly 50% of the total energy even in the mobile scenario.

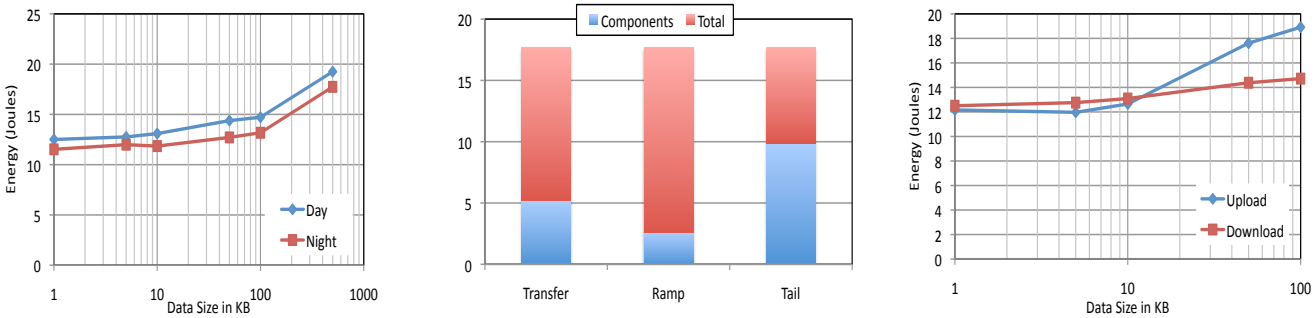
3.4 GSM Measurements

We conducted a set of measurements using the two Nokia phones equipped with GSM. Figure 6(a) shows the average energy con-



(a) Temporal variation: Tail energy (b) Temporal variation: Ramp energy (c) Geographical variation

Figure 3: 3G Measurements: 50K downloads (a) Tail energy over 3 days (b) Ramp energy over 3 days (c) Average tail and ramp energies measured in three cities using two devices D1 and D2.



(a) Download: Day versus Night (b) Upload measurements (c) Upload versus Download

Figure 4: 3G Measurements: (a) Comparison of average energy consumption of day versus night time downloads of 50K data.(b) Average transfer, tail and ramp energy consumed for uploading 50K data (c) Comparing the average energy consumed for downloading 50K data versus uploading 50K data.

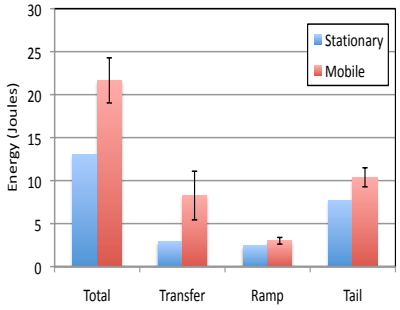


Figure 5: 3G measurements with mobility: Average energy consumed for downloading 50K of data.

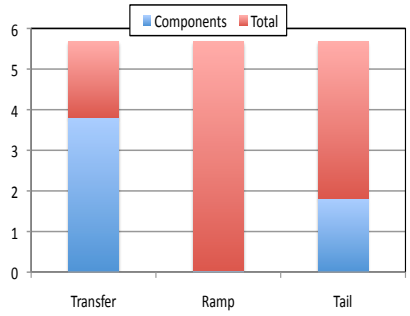
sumption in GSM networks as a proportion of the *Tail energy*, *Ramp energy* and transfer energy for a 50K download. Unlike in 3G, the *Tail energy* only accounts for 30% of the transfer energy. However, similar to 3G, the *Ramp energy* in GSM is small compared to the *Tail energy* and the transfer energy. We also observed that the *tail-time* is 6 seconds and GSM incurs a small maintenance energy between 2-3 J/minute (not shown in figure).

Due to the small *tail-time* in GSM (unlike 3G), data sizes dominate energy consumption rather than the inter-transfer times. Figure 6(b) shows the average energy consumed when varying the time between successive transfers. The average energy does not vary with increasing inter-transfer interval. For example, for data transfers of size 100 KB, the average energy consumption is between 19 Joules to 21 Joules even as the time between successive transfers is varied. In comparison, Figure 2(b) shows that the average energy consumption varies significantly in 3G with varying inter-transfer interval, until the inter-transfer interval grows to more than the *tail-time*.

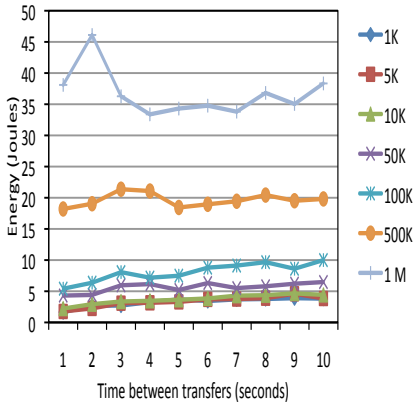
3.5 WiFi Measurements

Figure 7(a) shows the average energy consumption in WiFi composed of scanning, association and transfer, for a 50 K download. We observe that the scanning and association energy is nearly five times the transfer energy. Our results confirm previous measurements by Rahmati *et al.* [23].

Figure 7(b) shows that for WiFi, the energy consumption increases when time between successive transfer increases. Interestingly, energy consumption does not plateau after a threshold inter-transfer interval like in 3G (Figure 2(b)). The reason for increasing energy consumption with increasing inter-transfer interval is the high maintenance energy in WiFi. We measured the maintenance overhead (not shown) for keeping the WiFi interface on to be 3-3.5 Joules per minute.

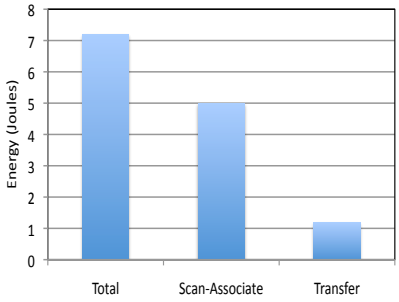


(a) GSM: Energy components

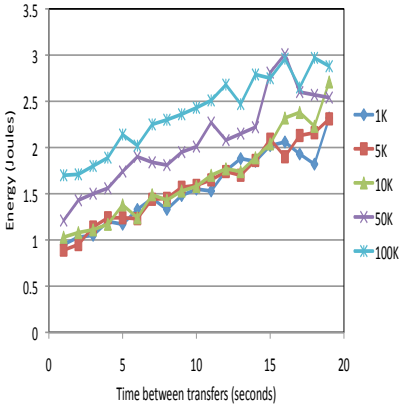


(b) GSM: Varying inter-transfer times

Figure 6: GSM Measurements: (a) Average ramp, transfer and tail energy consumed to download 50K data. The lower portion of the stacked columns show the proportion of energy spent for each activity compared to the total energy spent. (b) Average energy consumed for downloading data of different sizes against the inter-transfer time.



(a) WiFi: Energy components



(b) WiFi: Varying inter-transfer times

Figure 7: WiFi Measurements: (a) Average scan/associate and transfer energy consumed to download 50K data. The lower portion of the stacked columns show the proportion of energy spent for each activity compared to the total energy spent. (b) Averagenergy consumed for downloading data of different sizes against the inter-transfer time.

3.6 3G vs GSM vs WiFi

Figure 8 compares the average energy consumption of 3G, GSM and WiFi. 3G consumes significantly more energy to download data of all sizes (12-20J) compared to GSM and WiFi. GSM consumes 40% to 70% less energy compared to 3G. This is due to two reasons – (1) GSM radios typically operate at a lower power level than 3G radios and (2) the *Tail energy* set for GSM is around 6 seconds, much lower than the 12.5 seconds set for 3G.

WiFi is more energy efficient than both cellular networks once it is connected to an access point (AP). Figure 8 shows that the transfer energy for WiFi grows nearly three times slower compared to the cellular networks. For a download of size 10K, WiFi consumes one-sixth of 3G's energy and one-third of GSM's energy. With increasing data sizes WiFi's efficiency increases dramatically. The graph shows that when the cost of scan and transfer is included (marked in the graph as WiFi + SA), WiFi becomes inefficient for small sized transfers compared to GSM (as also observed by Rahmati *et al.* [23]). Surprisingly, when compared to 3G, WiFi is energy efficient even when the cost of scanning and association is included. We exploit this observation in Section 5.2.4.

Figures 9(a) and 9(b) show a qualitative comparison of the instantaneous power measurements over 3G and GSM respectively. The figure shows a snapshot of the power profile corresponding to a single 50K download. The lines marked *Uplink* and *Downlink* show the network activities in the uplink and downlink direction respectively (corresponding to request and download). The profiles show that GSM radio operates at lower power levels and it returns to the low power mode much earlier.

3.6.1 Energy model

One goal of our measurement study is to obtain accurate energy models for energy consumption over 3G, GSM and WiFi networks. The model enables us to empirically predict the energy consumption of different applications (Section 5). We model energy consumption as a function of both the size of transfer and the time between successive transfers. We note that both Figure 7(b) and Figure 2(b) show that time between transfers significantly affects energy consumption.

Table 1 tabulates the energy model we derive from the measurement study. The energy spent to download/upload x bytes of data over the cellular network consists of three components: 1) *Ramp energy* 2) transmission energy and 3) *Tail energy*. $R(x)$ denotes the sum of the *Ramp energy* and the transfer energy to send x bytes and E denotes the *Tail energy*. For WiFi, we use $R(x)$ to denote the sum of the transfer energy and the energy for scanning and association, and we set the *Tail energy* E to zero. In addition to the

		3G	GSM	WiFi
Transfer Energy	$R(x)$	$0.025(x) + 3.5$	$0.036(x) + 1.7$	$0.007(x) + 5.9$
Tail energy	E	0.62 J/sec	0.25 J/sec	NA
Maintenance	M	0.02 J/sec	0.03 J/sec	0.05 J/sec
tail-time	T	12.5 seconds	6 seconds	NA
Energy per 50KB transfer with a 20-second interval		12.5 J	5.0 J	7.6 J

Table 1: Energy model for downloading x bytes of data over 3G, GSM and WiFi networks. All values except the Maintenance values for 3G and GSM, are averaged over more than 50 trials.

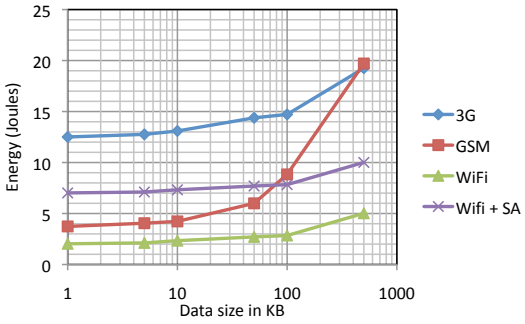


Figure 8: : WiFi versus 3G versus GSM measurements: Average energy consumed for downloading data of different sizes against the inter-transfer time

transfer cost, the total energy to transmit a packet also depends on the time the interface is on. M denotes the energy consumption to keep the interface on using M, the maintenance energy per second. Finally, T denotes the *tail-time*. The last row in Table 1 shows an example computation for the average energy spent to download 50K of data with a 20 second inter-transfer interval.

3.7 Measurements using HTC Fuze

We performed 3G and WiFi energy measurements from an HTC Fuze phone. For both experiments, we download a 50K byte packet 20 times over 3G and WiFi, with the inter-download interval set to 20 seconds. The measurements were performed in Redmond, WA. The 3G measurements are tabulated in Table 2. The average tail energy is 80% of the total energy consumption in HTC phones. In fact, the average tail time in Redmond is 14.3 seconds, 2 seconds more than the tail time in the Amherst-Boston area. The average transfer energy to download 50K data over WiFi is 0.92 J.

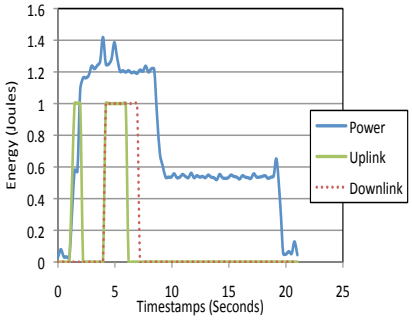
Avg tail energy	10.4 J
Avg tail time	14.3 s
Avg transfer energy	2.3 J

Table 2: 3G Measurement for downloading 50K data on HTC Fuze phone.

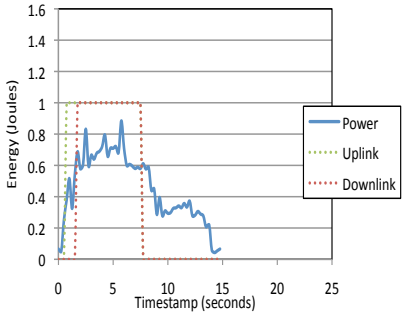
3.8 Summary

We summarize our key measurement findings as follows:

1. In 3G, nearly 60% of the energy is *tail energy*, which is wasted in high-power states after the completion of a typical transfer. In comparison, the *ramp energy* spent in switching to this high-power state before the transfer is small. The tail and ramp energies can be amortized over frequent successive transfers, but only if the transfers occur within *tail-time* of each other.



(a) 3G: Power Profile - 50K



(b) GSM: Power Profile - 50K

Figure 9: Power profiles of 3G and GSM networks

2. In GSM, although a similar trend exists, the *tail time* is much smaller compared to 3G (6 vs. 12 secs). Furthermore, the lower data rate of GSM implies that more energy is spent in the actual transfer of data compared to in the tail.

3. In WiFi, the association overhead is comparable to the tail energy of 3G, but the data transfer itself is significantly more efficient than 3G for all transfer sizes. When the scan cost is included, WiFi becomes inefficient for small sized transfers compared to GSM, but is still more energy efficient than 3G.

4. PROTOCOL

Informed by our measurement-driven model, we develop TailEnder, a protocol whose end-goal is to reduce energy consumption of network applications on mobile phones. Common network applications on mobile phones include e-mail, news-feed, software updates and web search and browsing. Many of these applications can be classified under two categories: 1) applications that can tolerate delays, and 2) applications that can benefit from prefetching. E-mail, news-feeds and software updates can be classified as applications that can tolerate a small user-specified delay. For example, a user may be willing to wait for a short time before e-mail and news-feeds are received, if it results in substantial energy savings.

For example, commodity phones such as the iPhone explicitly request the user to specify a delay-tolerance limit to improve battery life. Web search and browsing can be classified as applications that benefit from prefetching. Several studies [9, 16, 21] have shown that prefetching can significantly improve search and browsing experience for users.

TailEnder uses two simple techniques to reduce energy consumption for the two different classes of applications. For delay tolerant applications, TailEnder schedules transmissions such that the total time spent by the device in the high power state is minimized. For applications that can benefit from prefetching, TailEnder determines the number of documents to prefetch, so that the expected energy savings is maximized.

4.1 Delay-tolerant applications

First, we present a simple example to illustrate how applications can exploit delay tolerance to reduce energy utilization. Assume a user sends two emails within a span of a few minutes. The default policy is to send the emails as they arrive, and as a result the device remains in the high power state for two inactivity timer periods. However, if the user can tolerate a few minutes delay in sending the emails, the two emails can be sent together and the device remains in the high power state for only one inactivity timer period. Our measurement study shows that for low to moderate email sizes, the second strategy halves the energy consumption.

4.1.1 Scheduling transmissions to minimize energy

The goal of TailEnder is to schedule the transmission of incoming requests such that the total energy consumption is minimized and all requests are transmitted within their deadlines. We model the problem as follows. Consider a sequence of n requests where request i has an arrival time a_i and a deadline d_i by which it needs to be transmitted. Let $S = \{s_1, \ldots, s_n\}$ denote a transmission schedule that transmits request i at time s_i. The schedule S is *feasible* iff it transmits each request i after its arrival a_i and before its deadline d_i. When i is transmitted at time s_i, the radio transitions to the high power state, transmits i instantaneously, and remains in the high power state for T additional time units, where T is the *tail-time*. We ignore the relatively small energy overhead to switch to the high-power state. Even when multiple requests are transmitted at the same time, the device remains in the high power state only for T more time. Let $\phi(S)$ denote the *cost* or the the total time spent in the high-power state for the schedule S. The problem is to compute a feasible transmission schedule S that minimizes $\phi(S)$.

In practice, the request sequence is not known a priori, so we need to compute the schedule in an online manner. TailEnder uses the following simple idea: each incoming request is deferred until its deadline unless it arrives within $\rho \cdot T$ time of the most recent deadline when a request was scheduled, where $\rho \in [0, 1]$ is a constant that parameterizes the algorithm. Figure 10 presents the TailEnder algorithm for the scheduling problem. We prove the following two results to show that TailEnder is near-optimal.

THEOREM 1. *Any deterministic online algorithm is at best* 1.62*-competitive with the offline optimal algorithm.*

THEOREM 2. *TailEnder is* 2*-competitive with the offline optimal algorithm for any* $\rho \in [0, 1]$.

We provide a detailed proof of both theorems in the technical report [10] and outline the proof of Theorem 2 in the Appendix. Here, we outline the proof of Theorem 1. To this end, we describe a procedure to construct a request sequence that forces any deterministic online algorithm to incur a cost at least $1.62\times$ of the optimal.

TailEnder scheduler (t, r_i, d_i, a_i):

1. Let Δ be the last deadline when a packet was transmitted (initialized to $-\infty$ and reset in Step 3(c)).
2. If $(t < d_i)$
   (a) if $(\Delta + \rho \cdot T < a_i)$, transmit.
   (b) else add the request to queue Q.
3. If $(t == d_i)$
   (a) Transmit r_i
   (b) Transmit all requests in Q and set $Q = null$
   (c) Set $\Delta = d_i$

Figure 10: The TailEnder algorithm decides at time instant t whether to transmit a request r_i with arrival time a_i and deadline d_i. The parameter ρ is set to 0.62 in our implementation.

Let ALG be any deterministic online scheduler and ADV be the optimal offline scheduler. ADV observes the actions taken by ALG and generates subsequent requests. We show that no matter what scheduling decisions ALG makes, ADV can generate subsequent requests such that ALG incurs a higher total cost than ADV. For ease of exposition, we restrict ALG to only generate *nice* schedules. A *nice* schedule satisfies the following two properties: 1) it does not schedule any requests until the very first deadline; 2) it schedules each request immediately upon arrival or defers that request and subsequent requests until some deferred request's deadline is reached. We prove [10] that the restriction does not result in any loss of generality by showing that any schedule that is not nice can be converted to a nice schedule with equal or lower cost.

ADV generates requests as follows. ADV generates the first request at time 0 with a deadline also equal to 0. Thus, both ADV and ALG must schedule this request at time 0. Thereafter, ADV starts generating requests spaced infinitesimally apart all with a deadline $D \gg T$. Suppose the first request that ALG defers is a request r arriving at time xT.

- If ADV schedules request r and stops generating further requests, then $\phi(ADV) = (1 + x)T$ as ADV has no further requests to schedule. However ALG needs to schedule the request that arrived at xT, so $\phi(ALG) = (2 + x)T$. The cost ratio in this case is $\frac{2+x}{1+x}$, which decreases with x.
- If ADV defers all requests arriving after time 0 to the next deadline D, then D marks the end of the first epoch. Until D, $\phi(ADV) = T$ as ADV only scheduled once at 0 . However, $\phi(ALG) = (1 + x)T$. Thus, the cost ratio in the first epoch is $1 + x$, which increases with x.

The formal proof shows that the competitive ratio is at least the greater of the above two ratios, i.e., $\max(1 + x, \frac{(2+x)}{1+x})$, as ADV can make its decision after observing ALG's in each epoch. Thus, a lower bound on the competitive ratio is the minimum value of $\max(1 + x, \frac{(2+x)}{1+x})$. The corresponding value of x is obtained by solving $1 + x = \frac{2+x}{1+x}$ or the quadratic equation, $x^2 - x - 1 = 0$, which yields $x \approx 0.62$. Thus, the competitive ratio is at least 1.62.

4.2 Applications that benefit from prefetching

Our previous work [9] shows that *aggressive* prefetching can reduce response time for web search and browsing applications.

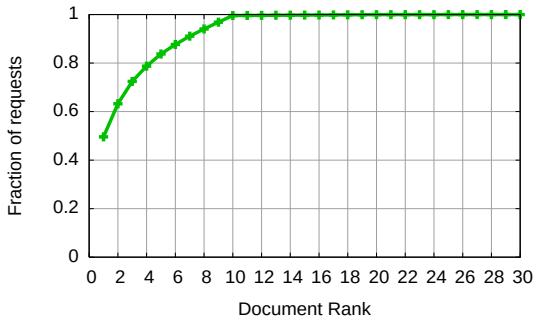


Figure 11: CDF of the fraction of times a user requests a web document at a given rank, for Web search application. The figure shows the CDF over more than 8 million queries collected across several days.

However, it is not straightforward to design a prefetching strategy whose end goal is to reduce energy consumption. On one hand, in the absence of any prefetching, the application needs to fetch the user-requested documents sequentially, incurring a large energy overhead. On the other, if the application aggressively prefetches documents and the user does not request any of the prefetched documents, then the application wastes a substantial amount of energy in prefetching. Clearly, predicting user behavior is key to the effectiveness of prefetching.

Our goal is to use user-behavior statistics to make prefetching decisions in the context of Web search and browsing. The information retrieval research community and search engine providers collect large amounts of data to study user behavior on the web. We model the prefetching problem as follows: *Given user behavior statistics, how many documents should be prefetched, in order to minimize the expected energy consumption?*.

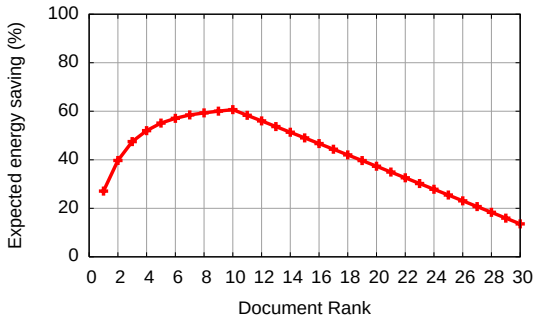


Figure 12: Expected percentage energy savings as a function of the number of documents prefetched.

4.2.1 Maximizing expected energy savings

Figure 11 shows the distribution of web documents that are requested by the user when searching the web. The graph is generated using Microsoft Search logs (obtained from Microsoft Live Labs). The logs contain over 8 million user queries and were collected over a month. Figure 11 shows that 40% of the time, a user requests for the first document from the list of *snippets* presented by the search engine. A user requests for a document ranked 11 or more, less than 0.00001% of the time.

We estimate the expected energy savings as a function of prefetched documents size. Let k be the number of prefetched documents, prefetched in the decreasing rank order and $p(k)$ be the probability that a user requests a document within rank k. Let E be the *Tail energy*, $R(k)$ be the energy required to receive k documents, and *TE* be the total energy required to receive a document. *TE* includes the energy to receive the list of snippets, request for a document from the snippet and then receive the document. For the sake of this analysis, we assume that user think-time to request a document is greater than the value of the inactivity timer. We do not make this assumption in our evaluation or the prefetching algorithm. The expected fraction of energy savings if the top k documents are prefetched is

$$\frac{E \cdot p(k) - R(k)}{TE} \quad (1)$$

Figure 12 shows the expected energy savings for varying k as estimated by Equation 1. The value of $p(k)$ is obtained from statistics presented in Figure 11, and E, $R(k)$ and *TE* are obtained from the 3G energy measurements (in Table 1). We set the size of a document to be the average web document size seen in the search logs.

Figure 12 shows that prefetching 10 web documents maximizes the energy saved. When more documents are prefetched, the cost of prefetching is greater than the energy savings. When too few documents are prefetched, the expected energy savings is low since the user may not request a prefetched document. Therefore, TailEnder prefetches 10 web documents for each user query. In Section 5, we show that this simple heuristic can save a substantial amount of energy when applied to real Web search sessions.

5. EVALUATION

We evaluate TailEnder using a model-driven simulation and real experiments on the phone. The goal of our evaluation is to quantify the reduction in energy utilization when using TailEnder for different applications, when compared to a Default protocol.

To show the general applicability of TailEnder, we evaluate its performance for three applications: emails, news-feeds and Web search. Email and news-feeds are applications that can tolerate a moderate delays; Web search is an interactive application but can benefit from prefetching. For all three applications, the impact of TailEnder for energy minimization largely depends on the application traffic and user behavior. For example, if a user receives an email once every hour, or if news-feeds are updated once per hour, TailEnder is unlikely to provide energy benefits. Therefore, we collect real application traces to evaluate TailEnder.

5.1 Application-level trace collection

For e-mail traces, we monitor the mailboxes of 3 graduate students for 10 days and log the size and time-stamps of incoming and outgoing mails. Table 3 tabulates the statistics of the resulting email logs. For news feed traces, we polled 10 different Yahoo!

	User 1	User 2	User 3
Incoming	446	405	321
Incoming size	214MB	162MB	161MB
Outgoing	219	183	354
Outgoing size	107MB	66MB	178MB

Table 3: Characteristics of collected e-mail

RSS news feeds[2] once every 5 seconds for a span of 3 days. We log the arrival time and size of each new story or an update to an existing story Table 4 lists the news-feeds we crawled. The traces cover major news topics, both critical (e.g., Business, Politics) and non-critical (e.g., Entertainment). Figure 5.1 shows the inter-arrival

[2]http://news.yahoo.com/rss

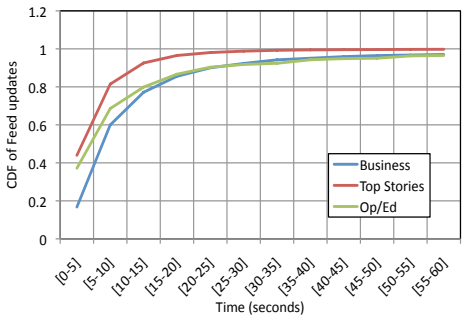


Figure 13: Arrival times distribution for 3 news feed topics.

times of updates for three example news topics – *Top stories, Op/Ed and Business*. Updates to the Top stories topic arrive with higher frequency than the Op/Ed and Business topics and nearly 60% of the updates for the Top stories topic arrive with an inter-arrival time of 10–15 seconds.

For Web search traces, we use the Microsoft Search logs that contains more than 8 million queries sampled over a period of one month. The search logs contain the clicked urls for each query and the time at which they were clicked. We extracted a random subset of 1000 queries from this data set. Figure 11 characterizes the distribution of clicks for each query.

Feed	Total stories
Opinion/Editorials	507
Health	2177
Technology	4659
Business	5616
Sports	7265
Politics	12069
US News	12389
Entertainment	16757
World	21006
Top Stories	23232

Table 4: News Feeds Collection

5.2 Model-driven evaluation

Our experimental methodology is as follows. The application trace consists of a sequence of arrivals of the form (s_i, a_i), where s_i is the size of the request and the a_i is the time of arrival. For example, for the news feed application, a_i is the time a topic is updated and s_i is the size of the update. Requests could be downloads as in news feeds or uploads as in outgoing emails. The Default protocol schedules transmissions as requests arrive. For delay tolerant applications, TailEnder schedules transmissions using the algorithm shown in Figure 10. For applications that benefit from prefetching, TailEnder schedules transmissions for all prefetched documents.

We estimate the energy consumption as a sum of the *Ramp energy* (if the device is not in high power state), transmission energy and the energy consumed because of staying in the high power state after transmission. If a request is scheduled for transmission before the *tail-time* of the previous transmission, the previous transmission does not incur an overhead for the entire *tail-time*. All results are based on the energy values obtained from day time, stationary measurements performed in Amherst, shown in Table 1.

5.2.1 News-feeds

Figure 14 shows the improvement in energy using TailEnder for each of the news feed topics. We set the deadline for sending the news feeds update to 10 minutes; i.e., a newsfeed content needs to be sent to the user with a maximum delay of 10 minutes since the content was updated. The average improvement across all news feeds is 42%. The largest improvement is observed for the Tech news feed at 52% and the smallest improvement for the Top story news feed at 36%. One possible reason for the top story news feed to yield lower performance improvement is that 60% of the top story updates arrive within 10 seconds, which is the less than the *tail-time* of 12.5 seconds (see Figure 5.1). Therefore, Default does not incur a *Tail energy* penalty for a large portion of the updates.

Figures 15 and 16 show the expected energy consumption for *business* news feeds using TailEnder and Default for varying deadline settings over 3G and GSM respectively. Figure 15 shows that as deadline increases to 25 minutes(1500 seconds), TailEnder's energy decreases to nearly half of the energy consumption of Default, decreasing from 10 Joules to 5 Joules per update. When sending data over GSM, the energy decreases from 6 Joules to 4 Joules when using TailEnder, compared to Default, yielding a 30% improvement.

5.2.2 E-mail

Figure 17 shows the energy reduction using TailEnder as percentage improvement over default, for incoming and outgoing emails for a set deadline of 10 minutes. We obtain an improvement of 35% on average for all three users. We note that we use download energy model for incoming emails and upload energy model for outgoing emails.

Figure 18 and Figure 19 show the energy consumed by TailEnder and Default as the deadline increases, when data is sent on the 3G and GSM networks respectively. This experiment is conducted using the incoming email traces of *User 2*. Increasing the deadline improves energy benefits of TailEnder in both 3G and GSM. Over 3G, TailEnder reduces energy consumption by 40% when the deadline is set to 15 minutes. As before, TailEnder provides a lower energy benefit in GSM compared to 3G.

5.2.3 Web search

Figure 20 shows the average per-query energy improvement using TailEnder for Web search application, when sending data over 3G and GSM. For Web search, TailEnder prefetches the top 10 documents for each requested query. Default only fetches documents that are requested by the user. TailEnder reduces energy by nearly 40% when data is sent over 3G and by about 16% when data is sent over GSM.

To understand the distribution of energy savings per query, we plot the CDF of the energy improvement in Figure 21. The plot shows that about 2% of the queries see little energy improvement. TailEnder reduces energy for 80% of the queries by 25–33%. For the remaining 18% of the queries, TailEnder reduces energy by over 40%. We find that the 18% of the queries that benefits most by TailEnder's prefetching are queries for which the user requested 3 or more documents.

5.2.4 Switching between 3G and WiFi

The decision to use either WiFi or 3G for data transfer involves, among other factors, a trade-off between availability and potential energy benefits. This is especially true when the user is mobile. While the WiFi interface consumes less energy per byte of transfer compared to 3G, the availability is less compared to 3G networks. One possible solution to get the energy benefits of WiFi but

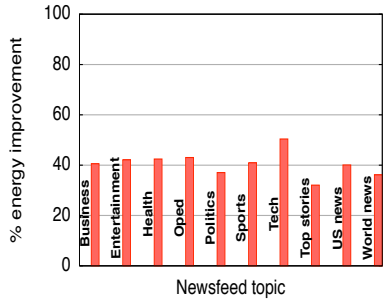


Figure 14: News feeds over 3G: Average per day energy improvement using TailEnder for different news feed topics

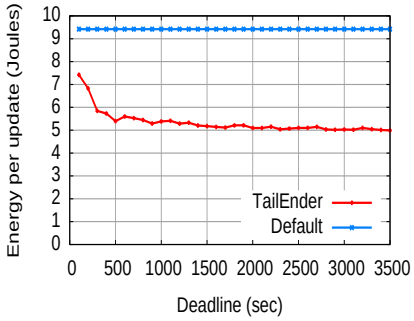


Figure 15: News feeds over 3G: Average energy consumption of TailEnder and Default for varying deadline

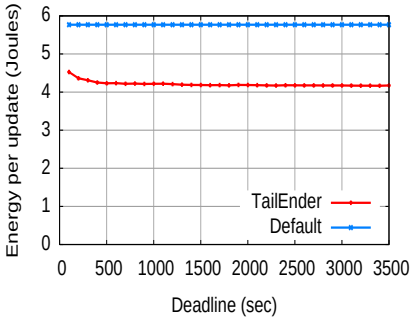


Figure 16: News feeds over GSM: Average energy consumption of TailEnder and Default for varying deadline

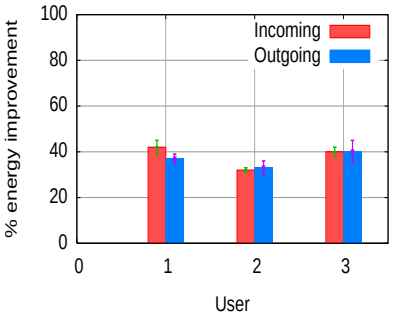


Figure 17: Email over 3G: Average per day energy improvement using TailEnder for email application

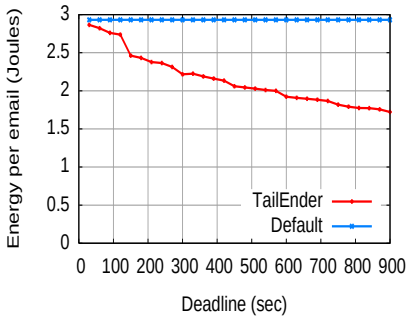


Figure 18: Email over 3G: Average energy consumption of TailEnder and Default for varying deadline

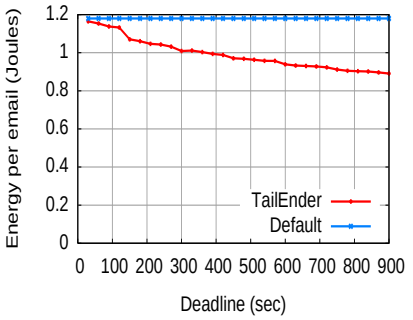


Figure 19: Email over GSM: Average energy consumption of TailEnder and Default for varying deadline

maintain availability, is to switch to the 3G interface when WiFi becomes unavailable. We conduct an experiment to upper bound the potential energy savings of switching between WiFi and 3G. Let WiFi be available only a fraction of the time. We assume that the WiFi interface is switched on only when WiFi is available, to avoid unnecessary scanning. Related work [23] show that WiFi availability can be predicted. We then estimate the energy savings in using the WiFi interface when available, and using 3G for the rest of the transfer.

Figures 22, 23 and 24 give an upper bound of energy benefit when switching between WiFi and 3G for news feed, email and Web search applications respectively. Keeping the WiFi interface on incurs a maintenance energy, as we observe in our measurement study (see Table 1). Therefore, in our experiment, we switch the WiFi interface off x seconds after a transfer if no data arrives. We estimate x as the ratio of the energy required for scanning/association and the per-second maintenance energy.

The figures show that when WiFi is always available, the energy consumption is 10 times lower compared to Default and more than 4 times lower compared to TailEnder for all three applications. Even when WiFi is available only 50% of the time, sending data over WiFi reduces energy consumption by 3 times compared to Default for all three applications. The results indicate that combining WiFi and 3G networks can provide significant energy benefits for mobile nodes without affecting network availability.

5.3 Experiments on the mobile phone

Next, we conduct data transfer experiments on the phone using the application-level traces. We convert an application trace into a sequence of transfers $S = \{< s_1, a_1 >, < s_2, a_2 >, \cdots, < s_n, a_n >\}$, such that data of size s_i is downloaded by the mobile phone at time a_i. Then, from a fully charged state, we repeatedly run this sequence of transfers until the battery drains completely.

We run two sequences of transfer, one generated by TailEnder and the other by Default. Given an application trace, TailEnder schedules the transfers according to whether the application is delay tolerant or can benefit from prefetching. Default schedules transfers as they arrive. We conduct the experiments for two applications: downloading Tech news feeds and Web search. For the news feed application, the metric is the number of stories downloaded and for Web search the metric is the number of queries for which all user requested documents were delivered.

Table 5 shows the results of the news feeds experiment. TailEnder downloads 60% more news feed updates compared to Default, and the total size of data downloaded by protocol increases from 127 MB to 240 MB providing a 56% improvement. Our model-based evaluation showed that for the Tech news feed, TailEnder can reduce energy by 52% compared to Default.

	Default	TailEnder
Stories	1411	3900
Total transfer size	127 MB	291 MB

Table 5: News feeds experiment. TailEnder downloads more than twice as many news feeds compared to Default on the mobile phone

Table 6 shows results for the Web search experiment. By prefetching, TailEnder sends responses to 50% more queries for the same

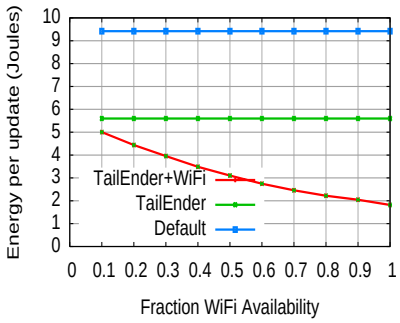


Figure 22: News feed. Average energy improvement when switching between WiFi and 3G networks

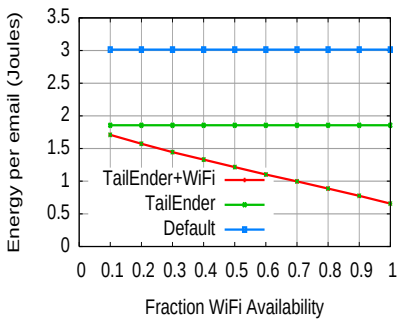


Figure 23: E-mail. Average energy improvement when switching between WiFi and 3G networks

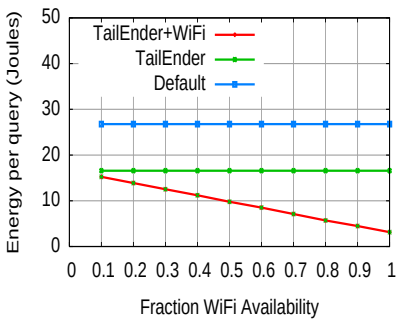


Figure 24: Web Search. Average energy improvement when switching between WiFi and 3G networks

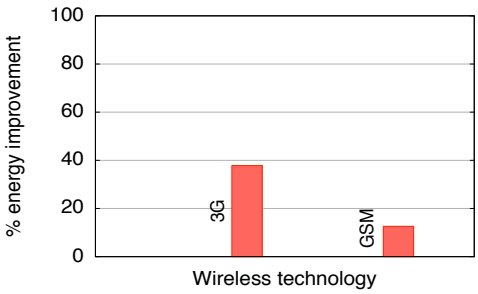


Figure 20: Web search: Average per query energy improvement using TailEnder for Web search over 3G and GSM

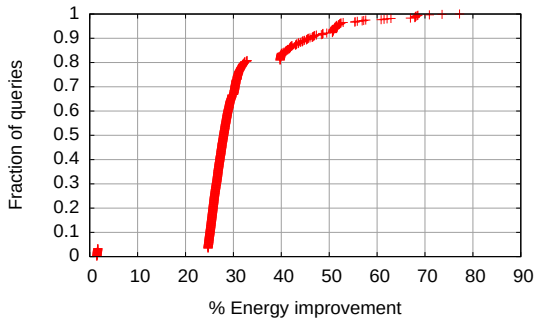


Figure 21: Web search: CDF of the energy improvement using TailEnder

amount of energy and the average number of transfers decreases by 45%. Prefetching is energy efficient, even though it sends ten times more data for each transfer on an average.

6. LIMITATIONS AND FUTURE WORK

TailEnder is naturally suited to be implemented in the operating system, exposing a simple API to applications. Applications only need to provide a delay-tolerance limit for each item sent. Today, commodity phones such as the iPhone already request the user to specify a delay-tolerance limit for certain applications in order to improve battery life. Implementing TailEnder in the kernel and refining the API to make it easily usable by users or application developers is left for future work.

Our results are based on email, rss feeds and web search traces collected from real desktop or laptop users. For a more realistic evaluation of TailEnder's energy savings however, we need the application usage patterns of users on mobile devices. Usage patterns on mobile devices provide two benefits. First, it helps quantify the energy benefits in the presence of cross-application optimization. For example, if a user multi-tasks between sending an email and searching the web, then the transmissions for the two activities can be scheduled together to reduce energy consumption. Second, the usage patterns provide us the fraction of time each application is used by a mobile user. This will help quantify the average per day energy savings for a given usage pattern. As part of future work, we seek to collect traces of mobile usage patterns that can inform cross-application opportunities and better quantify the energy benefits of TailEnder for mobile users.

	Default	TailEnder
Queries	672	1011
Documents	864	10110
Transfers	1462	1011
Average transfer sizes per query	9.3K	147.5K

Table 6: Web search experiment. TailEnder downloads 50% more queries compared to Default on the mobile phone

7. CONCLUSIONS

Energy on mobile phones is a precious resource. As phones equipped with multiple wireless technologies such as 3G, GSM, and WiFi become commonplace, it is important to understand their relative energy consumption characteristics. To this end, we conducted a detailed measurement study and found a significant tail energy overhead in 3G and GSM. We developed a measurement-driven model of energy consumption of network activity for each technology.

Informed by the model, we develop TailEnder, a protocol that minimizes energy usage while meeting delay-tolerance deadlines specified by users. For applications that can benefit from prefetching, TailEnder aggressively prefetches data, including potentially useless data, and yet reduces the overall energy consumed. We evaluate TailEnder for three case study applications—email, news feeds, and web search—based on real user logs and find significant savings in energy in each case. Experiments conducted on the mobile phone shows that TailEnder can download 60% more news feed updates and download search results for more than 50% of web queries, compared to using the default policy. Our model-driven simulation shows that TailEnder can reduce energy by 35% for email applications, 52% for news feeds and 40% for web search.

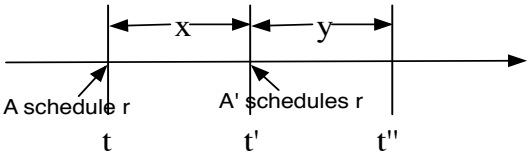


Figure 25: Construction for Lemma 2.

8. ACKNOWLEDGMENTS

This work was supported in part by NSF CNS-0845855 and the Center for Intelligent Information Retrieval at UMass Amherst. Any opinions, findings and conclusions or recommendations expressed in this material are of the authors and do not necessarily reflect those of the sponsors. The authors thank Rich Hankins from Nokia, Mark Corner, and Brian J Taylor for helping with the Nokia 95 devices used in this study. The authors also thank Rohan Murty for help with measurements in Boston, MA.

APPENDIX

In this section, we give formal proofs of the following results:

- *Lower bound*: Any deterministic online scheduling algorithm is at most 1.62-competitive compared to an offline optimal algorithm.
- *Upper bound*: TailEnder's cost is at most twice that of an optimal offline algorithm.

We use the following definitions and Lemmas to prove the bounds.

DEFINITION: The cost $\phi(S)$ of a schedule S is the time spent by the radio in the high power state when transmitting requests according to S. Let T denote the *tail-time*.

LEMMA 1. *Let A and A' be two algorithms with identical schedules except for the schedule of request r. Let A schedule r at t, when the device is the low-power state. If A' schedules r later, at $t' \geq t$ and if there are no other requests scheduled in $[t, t']$ then, $\phi(A) \geq \phi(A')$.*

PROOF. Let t'' be the first transmission after t' in A' (and after t in A by construction). Let $\phi(A_{[x,y]})$ denote the time the radio is in high-power states between times x and y due to A's schedule. Since A' is identical to A except for r, we must have $\phi(A_{[0,t]}) = \phi(A_{[0,t]})$ and $\phi(A_{[t'',t_e]}) = \phi(A'_{[t'',t_e]})$, where t_e is the last schedule.

Between t and t'', A is in the high power state for at most time $(t''-t)$ and A' is in the high power state for at most $(t''-t')$. Since $t \leq t'$ we have $(t''-t) \geq (t''-t')$. Thus, $\phi(A) \geq \phi(A')$. □

LEMMA 2. *Let A and A' be two algorithms with identical schedules except for the schedule of request r. Let A schedule r at t, when the device is the high-power state. IF A' schedules r later, at $t' \geq t$ and If there are no other requests scheduled in the interval $[t, t']$ then, $\phi(A) \leq \phi(A')$.*

PROOF. Let $t'' \geq t'$ be the farthest time after t' such that either a) the radio is in the high-power state throughout $[t', t'']$ or b) no requests are scheduled in the interval $[t', t'']$. In other words, after t'' either the radio returns to a low power state or a request is scheduled.

As before, since A' is identical to A except for r, we must have $\phi(A_{[0,t]}) = \phi(A'_{[0,t]})$ and $\phi(A_{[t'',t_e]}) = \phi(A'_{[t'',t_e]})$.

Let $(t'-t) = x$ and $t''-t = x+y$. Figure 25 shows times t, t' and t''. Recall that for A''s schedule device in the high power state through $x = [t, t']$ and by construction the device is in the high power state through $y = [t', t'']$. Therefore, A' is in the high power state from a duration of $x+y$.

Case 1: $x+y > T$: Since the distance between t and t'' is greater than T and A scheduled a request at t, the device will only be in the high power state for T time units i.e., $\phi(A_{[t,t'']}) = T$. Therefore, $\phi(A_{[t,t'']}) < \phi(A'_{[t,t'']})$

Case 2:. if $x+y \leq T$, we have $\phi(A_{[t',t'']}) = \phi(A'_{[t',t'']})$.

Therefore, together $\phi(A) \leq \phi(A')$. □

DEFINITION: A schedule is *nice* iff (1) it does not schedule any requests until the very first deadline; and (2) it schedules each request immediately upon arrival or defers that request and subsequent requests until some deferred request's deadline is reached.

LEMMA 3. *Any schedule that is not nice can be converted to a nice schedule of equal or lower cost. In particular, there exists a nice schedule with optimal cost.*

PROOF. Let A be the algorithm that generates a schedule that is not *nice*. Let request r be the first request such that A does not schedule r upon arrival. Request r arrives at time a, and the first deadline after its arrival is d. Let A schedule r at t such that $a < t < d$.

We construct an algorithm A' such that schedules generated by A and A' are identical except for the schedule of r. There are two cases:

Case 1: The device is in a low power state when r is scheduled.

Let A' schedule r and all subsequent requests at d. Then by Lemma 1, $\phi(A') \geq \Phi(A)$.

Case 2: Device is in high power state at time t when r was scheduled

- If the device remains in high power state from a to t: Let A' schedule r at its arrival a. Note that A schedules r at time $t > a$. By Lemma 2, $\phi(A') \leq \phi(A)$
- If the device is in low power state at a, or during any time in the interval $[a, t]$: Another request must have been scheduled at some time t' such that $a < t' < t$. Consider an algorithm A'' that is identical to A except that it schedules r at time t'. By Lemma 2, scheduling r at t' instead of t will create a schedule of equal or lower cost, $\phi(A'') \leq \phi(A)$. Further, at t' the device must have been in a low power state. Therefore, let A' schedule r and all subsequent requests at d (where $d \leq t'$). Using *Case 1*, we can show that $\phi(A') \leq \phi(A'')$. By transitivity, $\phi(A') \leq \phi(A)$.

Therefore, A' which schedules r *nicely* has equal or lower cost than A. Repeating the procedure for each request in A that is not *nicely* scheduled, we convert A into a *nice* schedule with equal or lower cost. Applying the procedure to any schedule with the *optimal* cost yields a *nice* schedule with the optimal cost. Hence, the Lemma.

□

Lower bound

LEMMA 4. *Any online algorithm that generates a nice schedule can at most be 1.62 competitive with an offline optimal algorithm.*

PROOF. Let ALG be any online algorithm that generates a *nice* schedule. We prove the theorem by constructing an offline adversary, ADV, that incrementally generates a request after observing the actions of ALG.

We divide time into epochs. In each epoch i, ADV begins by generating a request with a deadline d'_i. The arrival time of the first request in epoch i is after the deadline of the previous epoch. Let $d'_i - d'_{i-1} >> T$.

Consider epoch i. ALG has two choices.

Choice 1: If ALG schedules the current request, ADV generates a request after a small time ϵ. ADV continues to generate requests as long as ALG schedules requests upon arrival, but ADV does not schedule any of its requests.

Choice 2: If ALG defers the current request to the earliest deadline, ADV stops generating request for the epoch. Let the time that ALG stops generating request in the epoch i be x_i

Without loss of generality, let both ALG and ADV schedule their first request at time d'_0, the earliest deadline among all requests (from Lemma 3). Consider the beginning of the first epoch, at d'_0.

Case 1: If ALG schedules requests as they arrive until the end of the epoch, ADV schedules all requests at the epoch deadline d'_1. ADV only incurs a cost T, while ALG remains in the high power state through the entire epoch. The cost of ALG can be made arbitrarily large compared to ADV.

Case 2: If ALG defers requests at time $x_1 \leq 0.62T$. ADV generate no further requests and schedules all requests that were generated until x_1 immediately. Therefore, $\phi(ADV) = (1 + x_1)T$. On the other hand, ALG schedules the deferred request at the epoch deadline, so that $\phi(ALG) = (2 + x_1)T$. The competitive ratio is, $\frac{\phi(ALG)}{\phi(ADV)} = \frac{(2+x_1)}{(1+x_1)}$. For $x_1 \leq 0.62T$, the lower bound of $\frac{\phi(ALG)}{\phi(ADV)} = 1.62$, and the lower bound is achieved when $x_1 = 0.62T$. In this case, ALG is at best 1.62-competitive compared to ADV.

Case 3: If ALG defers requests at time $x_1 > 0.62T$. ADV generates no further requests in the current epoch and schedules all requests generated in the epoch at the epoch deadline. Therefore for the current epoch, $\phi(ADV) = T$ while $\phi(ALG) = (1+x_1)T$. Since $x_1 > 0.62T$, $\frac{\phi(ALG)}{\phi(ADV)} > 1.62$. ADV generates packets for subsequent epochs until the epoch i where ALG defers requests at time $x_i \leq 0.62T$.

- If such an i exists: For all epoch until $i-1$, $\frac{\phi(ALG)}{\phi(ADV)} > 1.62$. For epoch i, from *Case 2*, we have $\frac{\phi(ALG)}{\phi(ADV)} \geq 1.62$. Therefore the cost of ALG is at best 1.62-competitive compared to ADV.

- If no such i exists; i.e., ALG defers requests at times $x_i > 0.62T$ for all epochs: ADV stops generating requests after n epochs. For the first n epochs, $\phi(ALG) > 1.62T$ and $\phi(ALG) = T$. In the $n + 1$ epoch, both ALG and ADV incur a cost T to schedule pending requests from the previous epoch. Therefore, for all epochs together, $\frac{\phi(ALG)}{\phi(ADV)} = \frac{1.62n+1}{n+1}$. For n sufficiently large, $\frac{\phi(ALG)}{\phi(ADV)} \approx 1.62$.

From cases 1, 2 and 3, we show that any algorithm that generates a *nice schedule can at most be 1.62 compared to an offline adversary.* □

THEOREM 1 Any online algorithm can at most be 1.62 competitive with an offline optimal algorithm.

PROOF. Follows from Lemma 4 and Lemma 3. □

Upper bound

Here we prove the upper bounds on TailEnder's cost with respect to an optimal offline algorithm, using the following definitions and lemmas.Without loss of generality, we assume below that the *tail-time* is 1.

DEFINITION. Let *ARR* denote the nice schedule that begins scheduling at the very first deadline D_1 and schedules every request arriving after D_1 immediately upon arrival.

DEFINITION. Let *TEA* denote the nice schedule generated by TailEnder (in Figure 10) for some fixed $\rho \in [0, 1]$.

LEMMA 5. *The optimal cost of scheduling a request sequence does not increase by removing a subset of requests, or by increasing the deadlines of a subset of requests.*

LEMMA 6. *The cost of TEA is at most twice that of OPT, where OPT is an optimal nice schedule.*

PROOF. We prove the lemma using induction on the number of requests in the request sequence. It is easy to see that the lemma is true for all sequences of at most three requests. Consider an arbitrary request sequence, R, consisting of more than three requests.

Let OPT be an optimal x-schedule for this sequence. Let a denote the last request until which the schedules of both *TEA* and OPT are identical to *ARR*, i.e., they schedule every request at its arrival time. Let b be the first request where the *TEA* and OPT differ. Let D_1 denote the last deadline before a at which a request was scheduled (by both *TEA* and OPT). As illustrated in Figures 26 and 28, there are two cases 1) *TEA* schedules more requests than *OPT* or 2) *OPT* schedules more requests than *TEA*.

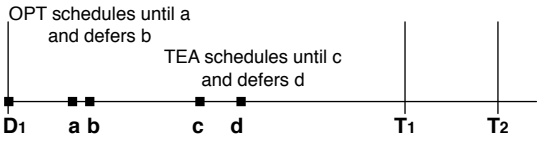


Figure 26: Illustration for Case 1.

Case 1. OPT defers b but *TEA* schedules until c and defers d. Let T_1 be the earliest deadline after b, and let T_2 be the earliest deadline after d. Clearly, $T_2 \geq T_1$ as shown in Figure 26. We complete the induction using a new request sequence.

Construct a new request sequence R' that is identical to R. If any request in the interval (D_1, T_1) has a deadline $< T_2$, change it to T_2. It is easy to verify that R' has the following properties:

- *TEA*ś schedule for R' is identical to its schedule for R.

- The cost of *OPT'*, an optimal schedule for R', is less than or equal to the cost of OPT, an optimal schedule for R.

OPT' has to start at D_1. Being an x-schedule, *OPT'* must defer at some request y after D_1 or remain identical to *ARR* until T_2. Note, if *OPT'* remains identical to *ARR* until T_2, *TEA* must have deferred at b which implies that we can use arguments from Case 2. Suppose *OPT'* schedules until x but defers some request y such that $D_1 \leq y < T2$. Then, there are two cases to consider as illustrated in Figure 27.

Case 1a. $y \leq d$.

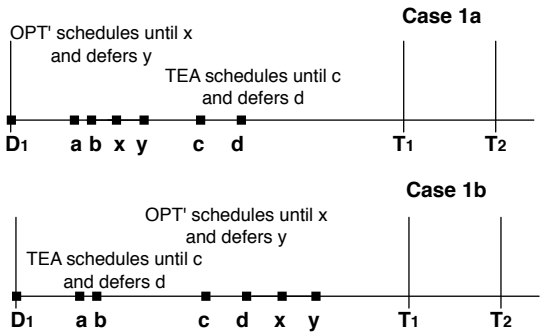


Figure 27: Illustration for Case 1a and 1b.

$$\begin{aligned}
\phi(\mathit{TEA}) &= \phi_{(0,D_1)}(ARR(D_1)) + \phi(ARR(D_1,c)) \\
&- \max(0, c+1-T_2) + \phi(\mathit{TEA}(|d)) \\
\phi(OPT') &= \phi_{(0,D_1)}(\mathit{ARR}(D_1)) + \phi(\mathit{ARR}(D_1,x)) \\
&\quad - \max(0, x+1-T_2) + \phi(\mathit{OPT}'(|y)) \\
&\geq \phi_{(0,D_1)}(\mathit{ARR}(D_1)) + \phi(\mathit{ARR}(D_1,x)) \\
&\quad - \max(0, x+1-T_2) + \phi(OPT'(|d)) \text{ (Lemma 5)} \\
\phi(OPT') &\geq \phi_{(0,D_1)}(\mathit{ARR}(D_1)) + \phi(\mathit{ARR}(D_1,x)) \\
&\quad - \max(0, x+1-T_2) + \phi(OPT(|d)) \text{ (Lemma 5)}
\end{aligned}$$

Since both *TEA* and *OPT*′ must schedule some request at D_1, in the interval (D_1, T_2), *TEA* can at most incur a cost of $1+\rho \leq 2$ and *OPT*′ has to incur at least 1. Therefore,

$$\phi(D_1,c) - \max(0, c+1-T_2) \leq 2\left[\phi(D_1,x) - \max(0, x+1-T_2)\right]$$

This implies,

$$\begin{aligned}
\phi(\mathit{TEA}) &\leq 2\phi_{(0,D_1)}(\mathit{ARR}(D_1)) + 2\phi(\mathit{ARR}(D_1,x)) \\
&\quad -2\max(0, x+1-T_2) + \phi(\mathit{TEA}(|d))
\end{aligned}$$

By inductive hypothesis, $\phi(\mathit{TEA}(|d)) \leq 2\phi(\mathit{OPT}(|d))$ which gives,

$$\begin{aligned}
\phi(\mathit{TEA}) &\leq 2\phi_{(0,D_1)}(\mathit{ARR}(D_1)) + 2\phi(\mathit{ARR}(D_1,x)) \\
&\quad -2\max(0, x+1-T_2) + 2\phi(\mathit{OPT}(|d)) \\
&\leq 2\phi(\mathit{OPT}') \\
\phi(\mathit{TEA}) &\leq 2\phi(\mathit{OPT}) \text{ (Lemma 5)}
\end{aligned}$$

Case 1b. If $y > d$ then *OPT*′ schedules more requests than *TEA* between D_1 and T_1 and therefore, Case 2 applies.

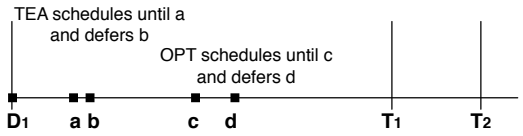


Figure 28: Illustration for Case 2.

Case 2. *TEA* defers b, but OPT schedules b. Suppose *TEA* defers b to the earliest deadline T_1 after b's arrival time. Suppose *OPT* is further identical to *ARR* until some request c and defers the next request d to the earliest deadline T_2 after d. Clearly, $T_2 \geq T_1$ as shown in Figure 28.

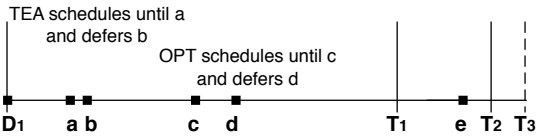


Figure 29: Illustration for Case 2a.

Case 2a. Suppose $T_2 - D_1 > 1+\rho$. Let T_3 be the next deadline to which *TEA* defers the first request e arriving after $T_1+\rho$ as shown in 29.

$$\begin{aligned}
\phi(\mathit{TEA}) &= \phi(\mathit{TEA}(a|)) + \phi(\mathit{TEA}(|b)) \\
&= \phi(\mathit{TEA}(a|)) + \phi(\mathit{TEA}(b,e)) + \phi(\mathit{TEA}(|e)) \\
&= \rho + \min(1, T_1 - D_1) \\
&\quad +\rho + \min(1, T_3 - (T_1+\rho)) + \phi(\mathit{TEA}(|e)) \\
&\geq \rho + 1 + \rho + 1 + \phi(\mathit{TEA}(|e)) \\
&\geq 2(\rho+1) + 2\phi(\mathit{OPT}(|e))
\end{aligned}$$

$$\begin{aligned}
\phi(\mathit{OPT}) &= \phi(\mathit{OPT}(c|)) + \phi(\mathit{OPT}(|d)) \\
&= \rho + x + \min(1, T_2 - D_1) + \phi(\mathit{TEA}(|d)) \\
&\geq \rho + x + 1 + \phi(\mathit{TEA}(|d)) \\
&\geq \rho + 1 + \phi(\mathit{TEA}(|d)) \\
&\geq \rho + 1 + \phi(\mathit{TEA}(|e)) \text{ (Lemma 5)} \\
\phi(\mathit{TEA}) &\leq 2\phi(\mathit{OPT})
\end{aligned}$$

Case 2b. Suppose $T_2 - D_1 < 1+\rho$. Both *TEA* and *OPT* must stay in the high-power state until T_2. Suppose *OPT* transitions to the low power state for the first time after D_1 at $x+1$ (by scheduling some request at x). Since *OPT* is nice, *OPT* defers all requests after x (starting at $y > x$) to the next earliest deadline τ_1.

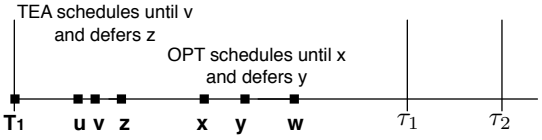


Figure 30: Illustration for Case 2b.

$$\begin{aligned}
\phi(\mathit{OPT}) &= \phi(\mathit{OPT}(x|)) + \phi(\mathit{OPT}(|y)) \\
&= x + 1 - D_1 + \phi(\mathit{OPT}(|y))
\end{aligned}$$

We make the following observations about *TEA*'s schedule.

- *TEA* must schedule some request between T_1 and x because there exists at least one request with a deadline T_1 that *TEA* deferred initially.

- Unless all requests beyond x arrive within ρ of their deadlines, *TEA* must defer some request after x. If all requests beyond x indeed arrive within ρ of their previous deadlines then, the best *OPT* can do on this request sequence is to schedule requests only in alternate epochs(as illustrated by the example in Figure 31). Thus, *TEA* will incur $\rho+1$ in all epochs but *OPT* will incur $1+\rho$ in every other epoch. Thus, $\phi(\mathit{TEA}) \leq 2\phi(\mathit{OPT})$.

Suppose u be the last request that *TEA* schedules before x and let w with a deadline of τ_2, be the first request after x that *TEA* defers.

$$\begin{aligned}\phi(\mathit{TEA}) &\leq \phi(\mathit{TEA}(u|)) + \phi(\mathit{TEA}(u,\tau_2)) + \phi(\mathit{TEA}(|w)) \\ &\leq u + 1 - D_1 + \phi(\mathit{TEA}(u,\tau_2)) + \phi(\mathit{TEA}(|w)) \\ &\leq x + 1 - D_1 + \phi(\mathit{TEA}(u,\tau_2)) + \phi(\mathit{TEA}(|w))\end{aligned}$$

If *TEA* schedules no requests in the interval (u, τ_2), $\phi(\mathit{TEA}(u,\tau_2)) = 0$. If *TEA* schedules some request, v, in the interval (u, τ_2), $\phi(\mathit{TEA}(u,\tau_2)) \leq 1 + \rho$. Note that $x - D_1 > \rho$. Therefore, $\phi(\mathit{TEA}(u,\tau_2)) \leq 1 + x - D_1$. This gives,

$$\begin{aligned}\phi(\mathit{TEA}) &\leq x + 1 - D_1 + x + 1 - D_1 + \phi(\mathit{TEA}(|w)) \\ &\leq 2(x + 1 - D_1) + 2\phi(\mathit{OPT}(|w)) \\ &\leq 2(x + 1 - D_1) + 2\phi(\mathit{OPT}(|y)) \text{ (Lemma 5)} \\ &\leq 2\phi(\mathit{OPT})\end{aligned}$$

Hence the proof. □

THEOREM 2 TailEnder is 2-competitive with the offline optimal algorithm for any $\rho \in [0, 1]$.

PROOF. Since TailEnder is a *TEA*-style algorithm, the proof directly follows from Lemma 6. □

THEOREM 3 TailEnder cannot be better than 2-competitive with the offline optimal algorithm for any $\rho \in [0, 1]$.

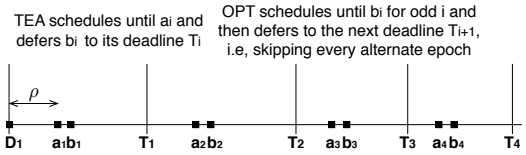


Figure 31: Illustration for showing that any *TEA*-style algorithm can be a factor 2 worse than optimal.

PROOF. We use the example sequence in Figure 31 to show that any *TEA*-style algorithm that defers ρ after that previous deadline for any $0 \leq \rho \leq 1$ can be a factor 2 worse than optimal. □

Thus, the upper and lower bounds on the competitive ratio for TailEnder-style algorithms are tight. However, bridging the gap between the $2\times$ upper bound and the $1.62\times$ lower bound for an arbitrary deterministic online algorithm remains an open problem.

A. REFERENCES

[1] 3g: Wikipedia. http://en.wikipedia.org/wiki/3G.

[2] International telecommunication union press release. http://www.itu.int/newsroom/press_releases/2008/29.html.

[3] Monsoon power monitor. http://www.msoon.com/.

[4] Third generation partnership project 2 (3gpp2). http://www.3gpp2.org.

[5] Third generation partnership project (3gpp). http://www.3gpp.org.

[6] Y. Agarwal, R. Chandra, A. Wolman, P. Bahl, K. Chin, and R. Gupta. Wireless wakeups revisited: energy management for voip over wi-fi smartphones. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 179–191, New York, NY, USA, 2007. ACM.

[7] T. Armstrong, O. Trescases, C. Amza, and E. de Lara. Efficient and transparent dynamic content updates for mobile clients. In *MobiSys '06: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 56–68, New York, NY, USA, 2006. ACM.

[8] J. Augustine, S. Irani, and C. Swamy. Optimal power-down strategies. In *FOCS '04: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science*, pages 530–539, Washington, DC, USA, 2004. IEEE Computer Society.

[9] A. Balasubramanian, B. N. Levine, and A. Venkataramani. Enabling Interactive Applications in Hybrid Networks. In *Proc. ACM Mobicom*, September 2008.

[10] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy Consumption in Mobile Phones: Measurement, Design Implications, and Algorithms. Technical Report 2009-22, UMass Amherst, 2009.

[11] P. Baptiste. Scheduling unit tasks to minimize the number of idle periods: a polynomial time algorithm for offline dynamic power management. In *SODA '06: Proceedings of the seventeenth annual ACM-SIAM symposium on Discrete algorithm*, pages 364–367, New York, NY, USA, 2006. ACM.

[12] X. Chuah, M.; Wei Luo; Zhang. Impacts of inactivity timer values on umts system capacity. In *Wireless Communications and Networking Conference (2002)*, volume 2, pages 897–903. IEEE, 2002.

[13] E. D. Demaine, M. Ghodsi, M. T. Hajiaghayi, A. S. Sayedi-Roshkhar, and M. Zadimoghaddam. Scheduling to minimize gaps and power consumption. In *SPAA '07: Proceedings of the nineteenth annual ACM symposium on Parallel algorithms and architectures*, pages 46–54, New York, NY, USA, 2007. ACM.

[14] A. Gupta and P. Mohapatra. Energy consumption and conservation in wifi based phones: A measurement-based study. In *Sensor and Ad Hoc Communications and Networks (SECON)*, pages 121–131, Washington, DC, USA, 2007. IEEE Computer Society.

[15] S. Irani, S. Shukla, and R. Gupta. Algorithms for power savings. In *SODA '03: Proceedings of the fourteenth annual ACM-SIAM symposium on Discrete algorithms*, pages 37–46, Philadelphia, PA, USA, 2003. Society for Industrial and Applied Mathematics.

[16] Z. Jiang and L. Kleinrock. Web prefetching in a mobile environment. In *IEEE Personal Communications*, volume 5, pages 25–34, September, 1998.

[17] R. Krashinsky and H. Balakrishnan. Minimizing Energy for Wireless Web Access Using Bounded Slowdown. In *8th ACM MOBICOM 2002*, Atlanta, GA, September 2002.

[18] C.-C. Lee, J.-H. Yeh, and J.-C. Chen. Impact of inactivity timer on energy consumption in wcdma and cdma2000. In *Proceedings of the Third Annual Wireless Telecommunication Symposium (WTS)*. IEEE, 2004.

[19] X. Liu, A. Sridharan, S. Machiraju, M. Seshadri, and H. Zang. Experiences in a 3g network: interplay between the wireless channel and applications. In *MobiCom '08: Proceedings of the 14th ACM international conference on Mobile computing and networking*, pages 211–222, New York, NY, USA, 2008. ACM.

[20] J. Nurminen, J.K.; Noyranen. Energy-consumption in mobile peer-to-peer - quantitative results from file sharing. In *Consumer Communications and Networking Conference (CCNC)*, pages 729–733, Washington, DC, USA, 2008. IEEE Computer Society.

[21] V. Padmanabhan and J. Mogul. Using Predictive Prefetching to Improve World Wide Web Latency. In *Proc. ACM Sigcomm*, pages 22–36, July 1996.

[22] T. Pering, Y. Agarwal, R. Gupta, and R. Want. Coolspots: Reducing the power consumption of wireless mobile devices with multiple radio interfaces. In *MobiSys 2006: Proceedings of the 4th international conference on Mobile systems, applications and services*, pages 220–232, New York, NY, USA, June 2006. ACM Press.

[23] A. Rahmati and L. Zhong. Context-for-wireless: context-sensitive energy-efficient wireless data transfer. In *MobiSys '07: Proceedings of the 5th international conference on Mobile systems, applications and services*, pages 165–178, New York, NY, USA, 2007. ACM.

[24] Y. Xiao, R. S. Kalyanaraman, and A. Yla-Jaaski. Energy consumption of mobile youtube: Quantitative measurement and analysis. In *NGMAST '08: Proceedings of the 2008 The Second International Conference on Next Generation Mobile Applications, Services, and Technologies*, pages 61–69, Washington, DC, USA, 2008. IEEE Computer Society.

[25] J.-H. Yeh, J.-C. Chen, and C.-C. Lee. Comparative analysis of energy-saving techniques in 3gpp and 3gp2 systems. In *Transactions*

on Vehicular Technology, volume 58, pages 432–448. IEEE, 2009.