A Packet Delaying Mechanism for Client-side Network Traffic to Reduce 3G Modem Related Energy Consumption of Mobile Devices

by

Erkut DEMİRHAN

A Thesis Submitted to the

Graduate School of Engineering

in Partial Fulfillment of the Requirements for

the Degree of

Master of Science

in

Computer Science and Engineering

Koç University

August, 2015

Koç University

Graduate School of Sciences and Engineering

This is to certify that I have examined this copy of a master's thesis by

Erkut DEMİRHAN

and have found that it is complete and satisfactory in all respects,

and that any and all revisions required by the final

examining committee have been made.

Committee Members:

Assoc. Prof. Dr. Serdar Taşıran (Advisor)

Prof. Dr. Alper Demir

Assoc. Prof. Dr. Özgür Gürbüz

Date: _____

To my family,

ABSTRACT

In modern mobile devices, 3G modems are one of the most energy consuming components. Therefore, utilization efficiency of 3G modems have a significant impact on battery life. Utilization of 3G modems is mostly determined by the 3G network usage patterns of mobile applications. However, most of the mobile applications are developed being unaware of energy consumption characteristics of 3G modems. They request 3G network connection for mostly small and periodic data transmissions. Each data transmission awakens the 3G modem and speeds up the battery consumption. This inefficient connection behaviour continues when the device is in the background mode (i.e., when the user is not interacting with the device).

In this thesis, we propose a method to reduce energy consumption of 3G modems in background mode. Our method implements a queueing mechanism to delay outgoing packet traffic for a predefined time and send them in batches. Thus, it reduces the total number of 3G modem wake-up events and hence the energy consumption.

We implement our method on a Motorola smartphone and collect background network traffic traces using well-known mobile applications. Using trace driven simulations, we demonstrate that our method can reduce 3G modem related energy consumption by up to 39% for different cases.

ÖZETÇE

Modern mobil aygıtlarda, 3G modemler en fazla enerji tüketimine neden olan bileşenlerdendir. Dolayısıyla, 3G modemlerin kullanım verimliliği batarya ömrünü önemli ölçude etkilemektedir. Bu verimlilik, önemli ölçüde mobil uygulamaların 3G ağ kullanma karakteristiği ile ilintilidir. Buna karşın, birçok mobil uygulama, 3G modemin enerji tüketim karakteristiği göz önünde bulundurulmadan geliştirilmektedir. Mobil uygulamalar, genellikle küçük ve periyodik veri transferleri için 3G ağ bağlantısı talep etmektedirler. Her veri transferi 3G modemi uyandırmakta ve batarya tüketimini hızlandırmaktadır. Bu verimsiz bağlantı kullanımı, mobil aygıt arka plan kipindeyken (kullanıcı aygıtla etkileşim halinde değilken) de devam etmektedir.

Bu tezde, 3G modemin arka plan kipinde harcadığı enerjiyi azaltmayı amaçlayan bir yöntem sunuyoruz. Yöntemimiz aygıttan çıkan ağ paketlerini belirli bir süre boyunca bir kuyrukta bekletmekte ve hepsini bir arada göndermektedir. Dolayısıyla, 3G modemin uyandırılma sayısını azaltarak enerji tüketimini düşürmektedir.

Yöntemimizi Motorola marka akıllı telefon üzerinde gerçekleştirdik ve arka plan kipinde popüler mobil uygulamaları çalıştırarak ağ paket trafiği izlerini topladık. Bu izler üzerinden benzetim yaparak enerji tüketimi değerlerini çıkardık ve farklı koşullar için enerji kullanımında %39'a varan azalma gözlemledik.

ACKNOWLEDGMENTS

First of all, I am deeply grateful to my advisor Assoc. Prof. Serdar Taşıran. His experience, understanding and patience added considerably to my graduate experience. I am also grateful to Prof. Alper Demir for his guidance and contribution throughout my thesis. I would also like to thank Assoc. Prof. Özgür Gürbüz for her precious time and being in my thesis committee.

I acknowledge the financial support of the Scientific and Technological Research Council of Turkey (TÜBİTAK) and facilities of Koç University during my M.Sc. study.

I would like to thank to present and former members of Multicore Software Engineering Research Center, Can, Burcu, Erdal, Hassan, İsmail, Maryam, and Suha, for their both technical and non-technical support. I would also thank Alper, Ayşegül, Bilgesu, Ceren, Doğus, Duygu, Engin, Ersan, Nazlı, Rustam and Tayfun for their valuable friendships. Last but not the least, I cannot forget Deniz Demircioğlu, who is no different than my biological brother after 8 years of being schoolmate, colleague, and housemate.

Finally, I would like to express my endless gratitude to my parents Ertan and Alev Demirhan, my brother Ekrem, and my sister İlayda, for their unconditional love and

support. I am dedicating this thesis to them.

TABLE OF CONTENTS

List of Tables **xii**

List of Figures **xiii**

Nomenclature **xv**

Chapter 1: Introduction **1**

1.1 Motivation 1

1.2 Approach 3

1.3 Contributions 3

1.4 Organization of the Thesis 4

Chapter 2: Power Consumption in 3G Networks **5**

2.1 The RRC State Machine 5

2.2 Previous Research 7

Chapter 3: Packet Traffic Analysis **11**

3.1 Data Collection 11

3.1.1 Devices and Tools 11

3.1.2 The Application Set 12

3.1.3 Methodology 13
3.2 Analysis 14
3.2.1 Packet Sizes 14
3.2.2 Packet Bursts 14
3.2.3 Pull Based vs. Push Based Transmissions 19
3.2.4 Transport Layer Analysis 21

Chapter 4: Packet Delaying Method 23

4.1 System Design 23
4.2 Packet Delaying Algorithm 25
4.2.1 Excluding Identical Packets from the Buffer 27
4.3 Implementation 28

Chapter 5: Evaluation 30

5.1 Experimental Setup 30
5.1.1 Application Set 31
5.1.2 Trace Driven Simulation 31
5.2 Comparison of Different Packet Delay Periods 32
5.3 Impacts of the Packet Delaying Method on Different Application Categories 39
5.4 CPU Overhead of Running the Packet Delaying Method 41

Chapter 6: Conclusion 43

Bibliography **44**

LIST OF TABLES

3.1 Average number of bursts per trace, and packets per burst for each application and all applications in the test set 16

3.2 Percentages of TCP and UDP packets 21

3.3 Percentages of TCP traffic analysis categories 22

5.1 RRC state machine parameters for trace driven simulation 33

5.2 Mean and median amounts of energy consumption for different packet delay periods 34

5.3 Percentages of different TCP packet categories over total number of TCP packets, for different delay settings 38

5.4 Application categories and corresponding applications 40

LIST OF FIGURES

2.1 The RRC state machine 6

3.1 Cumulative packet size distribution for the application set 15

3.2 Cumulative distribution of inter-burst times for each application . . . 17

3.3 Uplink and downlink packet traffic of sample data collection sessions . 18

3.4 Percentage 3G modem wake-up events due to uplink packets for each individual application, and all applications combined 20

4.1 System design of the packet delaying method 25

5.1 Percentages of (a) total packet delay time over total trace durations, and (b) delayed packets over total number of uplink packets for each delay period 35

5.2 Cumulative distributions of (a) energy consumption and (b) number of packets in all traces for different packet delay periods 36

5.3 Reduction in the number of bursts for different packet delay periods . 37

5.4 Percentage of energy savings for different application categories . . . 40

5.5 Reduction in the number of bursts for different application categories 41

5.6 Cumulative distribution of the CPU utilization percentage of the packet delaying method for all traces 42

NOMENCLATURE

$3GPP$	Third generation partnership project
CN	Central network
CPU	Central processing unit
DNS	Domain name system
$HSPA+$	Evolved high-speed packet access
$ICMP$	Internet control message protocol
IP	Internet protocol
RNC	Radio network controller
RRC	Radio resource control
TCP	Transmission control protocol
UE	User equipment
UDP	User datagram protocol
$UMTS$	Universal Mobile Telecommunications Systems

Chapter 1

INTRODUCTION

1.1 Motivation

Smartphones, tablets and other mobile devices have been integrated into our everyday lives. They have become one of the most common means of accessing the internet. In fact, mobile connectivity has increased so much in recent years that, it has even surpassed connection through desktop PCs. According to the 2014 survey of International Telecommunication Union, by the end of 2014, the number of mobile devices that subscribed to a cellular network has almost reached 7 billion [18]. Moreover, increase in the performance of mobile device components, such as CPUs and cellular modems, has enabled development of more complex mobile applications that demand high data transmission and computation capabilities.

Contrary to all of the advances stated above, the battery has continued to be the scarcest resource in mobile devices. Battery life limitation has been restraining mobile device users from fully benefiting from the features of their devices. Using functionalities, such as GPS navigation, web browsing, video streaming, that a typical mobile device provides, can drain the battery in a blink of an eye. Thus, battery

efficiency has been remaining as one of the most profound concerns in the mobile world.

3G modems are one of the most energy consuming mobile device components. Most of the mobile device users are aware of this phenomenon by experience. In order to reduce the battery consumption due to 3G network usage, many web-sites and blogs on mobile technology recommend methods such as disabling 3G modem, or changing notification settings to reduce the amount of data retrieved. Even though these methods both limit the user experience and contradict with the "always on" purpose of cellular network connectivity, they still seem as the only feasible options users can choose.

3G modems' high energy consuming nature increases the importance of their utilization efficiency. Pathak et al. [13] find that, after data transmission finishes, 3G modems continue consuming the same amount of power for an extended period of time (which is called "tail time", or "tail period" in literature). Another study [3] shows that, 60% of the total energy spent by the 3G modem is wasted during tail periods. In other words, 60% of the energy is spent when there is not any data transmission. Thus, the number of tail periods a 3G modem enters is a defining factor for energy consumption. On the other hand, most of the mobile applications are developed being unaware of this fact. They request cellular network connection even for small data transmissions and increase the number of tail periods. Moreover, a typical mobile device includes multiple running applications. Since these applications use the cellular network being unaware of each other, they can increase the inefficiency in 3G

modem utilization.

Many mobile applications continue data transmission during the mobile device is in the background mode for performing tasks such as notification checking, status updating and advertisement downloading. So, this situation perpetuates the problem mentioned above, even during the background mode.

1.2 Approach

Key intuition in our approach is to monitor uplink (from the device) and downlink (to the device) cellular network activity of mobile applications during the mobile device is in the background mode and to delay uplink network packets during appropriate time intervals. We perform delaying by routing uplink packets into a buffer and keeping them for a predefined period before sending. The reason we focus only on the background traffic is to prevent degrading user experience. The foreground traffic can include delay intolerant packets coming from user interactive tasks, such as web browsing, video streaming or voice-over-IP calling.

We aim to reduce the number of times 3G modem wakes up for data transmission and hence to reduce the amount of energy spent due to small data transfers in background traffic. We also consider not to reduce application performance and increase signaling overhead.

1.3 Contributions

This thesis has the following contributions:

• We provide an analysis of cellular network traffic of mobile devices in background mode, using real data traces (collected from Motorola smartphone in AVEA's 3G network). Based on this analysis, we show applicability of our approach.

• We propose a method that monitors uplink and downlink packet activity and shapes background network traffic to reduce 3G modem related energy consumption. Our method includes an algorithm to aggregate uplink network packets by delaying them for a predefined period.

• We introduce an Android application implementation that leverages Linux netfilter module to manipulate background network traffic. Our implementation reduces 3G network usage related energy consumption without requiring any modification to the application code.

1.4 Organization of the Thesis

In Chapter 2, we provide the necessary background information and discuss related studies on the topic. In Chapter 3, we provide our analysis on the cellular network traffic generated by mobile applications running in the background. In Chapter 4, we introduce the packet delaying method. In Chapter 5, we provide our experimental results and evaluate performance of the packet delaying method. In Chapter 6, we make concluding remarks, and discuss future work.

Chapter 2

POWER CONSUMPTION IN 3G NETWORKS

In this chapter, we provide necessary background information about 3G network technologies we focus throughout our study. We explain the interaction between mobile devices and 3G network providers to allocate and maintain network resources.

In addition to the background information, we also provide related studies on increasing energy efficiency in 3G networks.

2.1 The RRC State Machine

In this thesis, we refer to the Universal Mobile Telecommunications System (UMTS) as the cellular network technology. The UMTS is one of the most commonly used third generation (3G) mobile cellular communication technologies [8].

In the UMTS network, base stations implement the radio resource control (RRC) protocol that manages connectivity between mobile devices and network operators. To efficiently allocate radio resources, the RRC protocol operates a state machine with each subscribed mobile device, which is shown in Figure 2.1 [3].

The RRC state machine includes three states: Cell_DCH, Cell_FACH and IDLE. In each of these states, the 3G modem shows different power consumption and data transmission characteristics. In the IDLE state, the 3G modem does not perform any

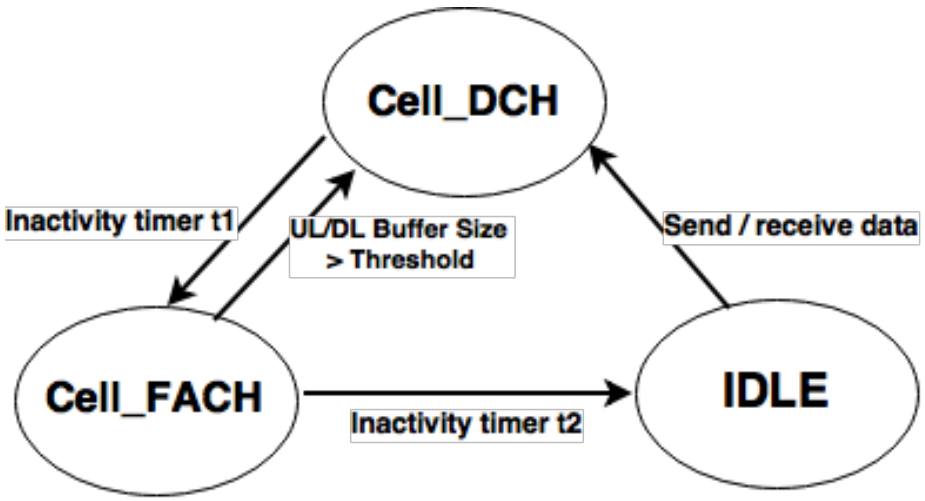


Figure 2.1: The RRC state machine

network activity. However, in Cell_DCH and Cell_FACH states, it is actively involved in data transmission. During the Cell_DCH state, mobile devices fully utilize the whole network channel themselves. They achieve the highest data throughput with the lowest latency. In the Cell_FACH state, multiple mobile devices share the same network channel with lower data throughput compared to the Cell_DCH state.

3G modems consume the highest amount of power during the Cell_DCH state. During the Cell_FACH state, 3G modems consume about 50% of power they consume in Cell_DCH state. In IDLE state, 3G modems consume about 1% of power they consume during Cell_DCH state [3].

State transitions of the RRC state machine can be divided into two groups: state promotions (IDLE→Cell_DCH and Cell_FACH→Cell_DCH) and state demotions (Cell_DCH→Cell_FACH and Cell_FACH→IDLE). State promotions switch the cellular modem to a higher power consuming and cellular resource utilizing state.

State demotions go in the reverse direction.

During state promotions, mobile devices wait for the allocation of cellular network resources before starting data transmissions. In this period, radio resource control messages are transferred between the mobile devices and the network providers. During IDLE state, sending or receiving any data triggers IDLE→Cell_DCH promotion. In Cell_FACH state, if uplink (UL) or downlink (DL) data throughput exceeds the predefined threshold, the RRC state machine switches to the Cell_DCH state.

If the number of state promotions increases, signaling overhead and delays occur. Thus, the user experience degrades. To balance the number of state promotions, the RRC state machine implements inactivity timers before each state demotion: t_1 for Cell_DCH→Cell_FACH, and t_2 for Cell_FACH→IDLE. If any data activity occurs before inactivity timers finish, inactivity timers are restarted. Sum of time bounds of these inactivity timers is called "tail time", or "tail period" in literature.

Even though tail periods reduce the number of state promotions, they also incur significant amount of energy consumption, as they postpone switching to a lower power consumption state. Therefore, there is a trade-off between state promotion overhead and energy consumption.

2.2 Previous Research

There have been various studies on analysis and reduction of cellular network related power consumption in mobile devices.

Some studies utilize a 3GPP specification called fast dormancy [5] to reduce the

time spent during tail periods. The fast dormancy mechanism enables mobile devices to notify cellular network by sending a special radio controlling message, and immediately release allocated radio resources. In other words, the fast dormancy feature triggers immediate transition to IDLE state. Authors of tail optimization protocol (TOP) [15] propose an API for mobile applications to send their predictions for their next data transmission, and using these predictions, the protocol decides when to apply fast dormancy. This approach requires adjustments to the applications to use the API. Moreover, how each application makes prediction is not clear enough. Authors of RadioJockey [1] propose a learning method that uses end of network communication points of applications' system call traces and predicts for suitable periods to apply fast dormancy. However, since the method needs entire system call traces of all applications to make accurate predictions, it incurs a high amount of computational overhead. Authors of SmartCut [19] propose a method that uses temporal correlations of historical packet trafic data to predict arrival times of future transmissions. Using these predictions, the method cuts unnecessary tails by applying fast dormancy. The method needs to regularly update its prediction model to adapt changes in user behaviour and network condition. This requirement increases the computational overhead.

Another group of studies focus on tail period sharing. Tail period sharing is a method that schedules delay tolerant network traffic to tail periods of the remaining network communications. Therefore, it increases the utilization of tail periods. Authors of TailEnder [3] introduce a method to schedule delay tolerant outgoing net-

work data while meeting user defined deadlines. TailEnder also prefetches data for applications that can benefit from prefetching. Liu et al. [10] propose an algorithm, called TailTheft, to separate the network traffic that can be prefetched or delayed from the rest, and to schedule them for the tail periods of other transfers. Lagar et al. [9] propose a technique, called traffic backfilling to enable applications to separate their delay tolerant data transmissions from the rest, and to reschedule delay tolerant data in between bursts of interactive packet traffic to increase utilization of cellular resources. Another study [4] proposes an adaptive online scheduling algorithm to improve energy efficiency while also considering user experience concerns.

Yeh et al. [20], and Falaki et al. [6] study effects of adjusting tail period timers on energy consumption and radio resource utilization. Although lowering tail periods cause RRC state promotion overhead, Falaki et al. show that reducing tail period duration from its default value, which is measured in their study as 17 seconds, to 4.5 seconds do not increase the number of state promotions for 95% of the network traffic they studied.

There have also been studies that focused on profiling 3G network resource usage. Qian et al. [16] designed an inference method for RRC state machine using traces collected from mobile devices. They implemented a tool that uses the inference method and network packet traffic traces to analyse energy consumption of mobile applications. They observed that even the most popular applications can have energy inefficient traffic patterns due to small data transfers, disorganized prefetching and unnecessary refreshes. The same research team also studied periodicity [14]

of application traffic in 3G network. They propose a method for defining periodic data transmissions. They investigated both origins and effects of periodic transfers, and found that periodic transfers are generated by reasons such as keep-alive, polling and user behaviour measurements. They also found that even though periodic transfers account for a tiny fraction of packet traffic, they cause huge amount of energy consumption.

Chapter 3

PACKET TRAFFIC ANALYSIS

To understand characteristics of the 3G network traffic that applications generate in background mode and to see its viability to apply our approach, we make an analysis of background network traffic generated by popular mobile applications.

In our analysis, using network packet traces we collect, we study packet size and burst distributions, pull-based and push-based data transmission behaviours and transport layer characteristics.

3.1 Data Collection

For this analysis, we collect network packet traces using the TCPDUMP program. Each trace includes network packet information including packet sizes, packet arrival times, internet protocol (IP) and transport layer protocol header information.

3.1.1 Devices and Tools

We performed the data collection using a Motorola Razr-i XT890 smartphone. Properties of the device are listed below:

Device Properties 2GHz single-core Intel Atom Z2460 Processor, 1 GB RAM, 2000mAh Battery, 3G modem, Running Android 4.1.2 (Jelly Bean) with superuser

permissions

We perform data collection in 3G network of the mobile phone operator AVEA that uses HSPA+ technology. To collect packet traces, we use AT& T's application resource optimizer (ARO) [2]. ARO is a free cross-platform tool for optimizing mobile application performance. It provides a mobile application to collect data from mobile devices for performance analysis. The application includes a TCPDUMP program for gathering network packet traces. To be able to run the application on our device, we cross-compiled the TCPDUMP source for Intel x86 architecture, and integrated its executable into the application.

3.1.2 The Application Set

We use the following well-known applications for the analysis. We select these applications as they generate packet traffic during the background mode.

Facebook: A well-known social media application. We use an account having around 200 connections. We set the application's notification checking period as 30 mins., which is the most frequent option available.

GMail: An e-mail service application provided by Google. We use one mail account. The application runs a background process, and checks server for new e-mail every 30 mins.

Skype: A voice over IP and instant messaging application. We use a Skype account having around 20 connections. The application runs a background process checking for an incoming message or call request.

Twitter: A well-known social media application. We use an account that follows updates of 52 popular accounts from different locations of the world, and is followed by 25 other accounts. We set the application's notification checking period as 5 mins., which is the most frequent option available.

Since the mobile application traffic in background mode does not include any network activity resulting from user interaction, we can assume that mobile applications generate similar network traffic patterns regardless of user. Therefore, even though we collect packet traces of applications belong to only one user, these traces can represent background network traffic of all users.

3.1.3 Methodology

We perform data collection in different periods of the day starting from 8 AM to 10 PM. Before gathering traffic data of an application or a group of applications, we stop all of the other third party applications running on the device. A typical data collection routine includes starting the target application(s) and the ARO data collector, closing the screen, keeping the screen off for 2 hours and then stopping the ARO data collector and retrieving network packet traces.

As we orient our analysis on the network activity in the background mode, we keep the device's screen off during data collection. In order to prevent the network packet traffic affected by signal power variations and base station switches, we also keep the testing device in the same location.

We collect 9 network packet traces for each individual application, and combina-

tion of all applications in the test set, which makes 45 traces in total. Each trace is 2-hour-long, which is stated as long enough for observing traffic patterns in a previous study on periodic network data transfers of mobile applications [14].

3.2 Analysis

3.2.1 Packet Sizes

As Falaki et al. [6] state in their study, small sized packet transfers imply inefficiency in energy consumption.

To see the contribution of small sized packet transfers to the background network traffic, we analyse the packet size distribution of all traces we collected. Figure 3.1 shows cumulative distribution of packet sizes in traces belong to each application, and all applications combined. As seen in the figure, for all cases, more than 50% of packets have size of less than 100 bytes. This result shows that significant percentage of background network traffic is generated by small packet transmissions.

3.2.2 Packet Bursts

A packet burst is a group of consecutive packets with relatively small inter-arrival times (i.e., time between arrival of two consecutive network packets). Number of bursts in a packet traffic, and number of packets in each burst are among the most defining factors of 3G modem utilization efficiency. Because they determine the duration in which 3G modem stays in the high power consuming state.

In this section, we make a burst analysis on the data traces we collected. In our

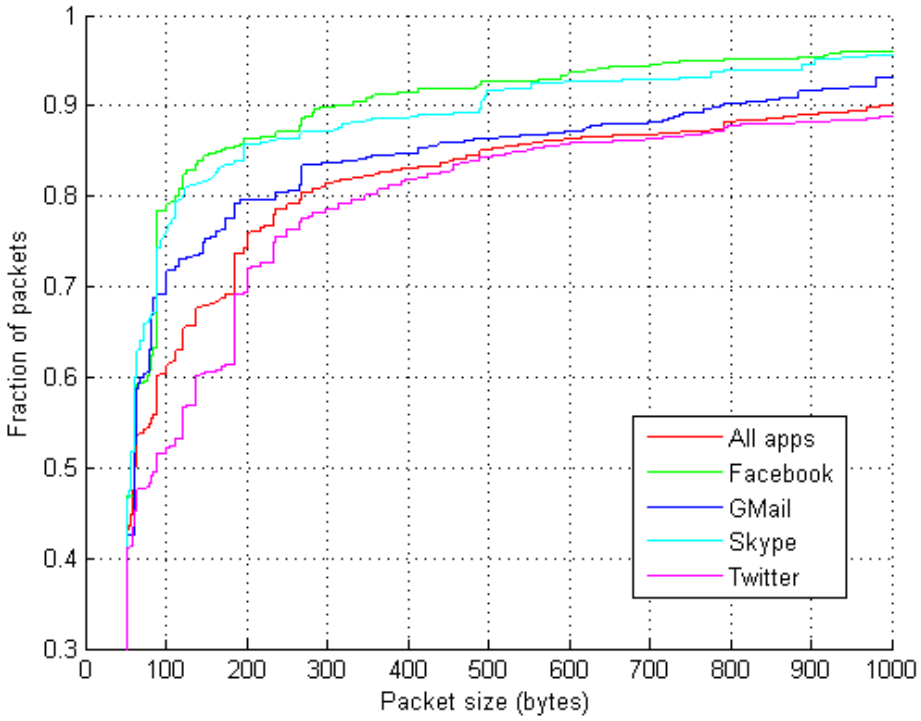


Figure 3.1: Cumulative packet size distribution for the application set

analysis, we use the burst definition introduced in a previous study [16] that analyses mobile application traffic. The definition is as follows:

Let $< t_0, t_1, ..., t_k >$ be the arrival times of the consecutive packets $< p_0, p_1, ..., p_k >$, and δ be a given threshold value. Then, for some i, j, such that $0 \leq i \leq j \leq k$, $< p_i, ..., p_j >$ is a burst iff:

- $t_m - t_{m-1} \leq \delta$ for all $i < m \leq j$,
- $t_i - t_{i-1} > \delta$ if $i > 0$, and
- $t_{j+1} - t_j > \delta$ if $j < k$.

Similar to the burst definition in [16], we set the δ as 1.5 seconds.

We analyse the number of bursts per trace, and the number of packets per burst for applications in the test set, and obtain the results in Table 3.1. As we discussed above, high number of bursts with small quantity of packets is a sign of 3G modem utilization inefficiency. We observe that characteristic in all traces except for the ones belong to GMail.

Table 3.1: Average number of bursts per trace, and packets per burst for each application and all applications in the test set

Application	Avg. number of bursts per trace	Avg. number of packets per burst
All applications	102	16
Facebook	64	9
GMail	16	16
Twitter	92	13
Skype	93	18

In addition to the number and length of bursts, time spent between consecutive bursts (i.e., inter-burst time) is also an important factor on determining the efficiency of 3G modem utilization. Thus, we also analyse inter-burst time distribution of traces for each application. Being consistent with our burst definition, we define the inter-burst time as any packet inter-arrival time higher than δ. Figure 3.2 shows the cumulative distribution of inter-burst times for each application and all applications. All traces, except the ones belong to GMail application, have similar inter-burst time distributions.

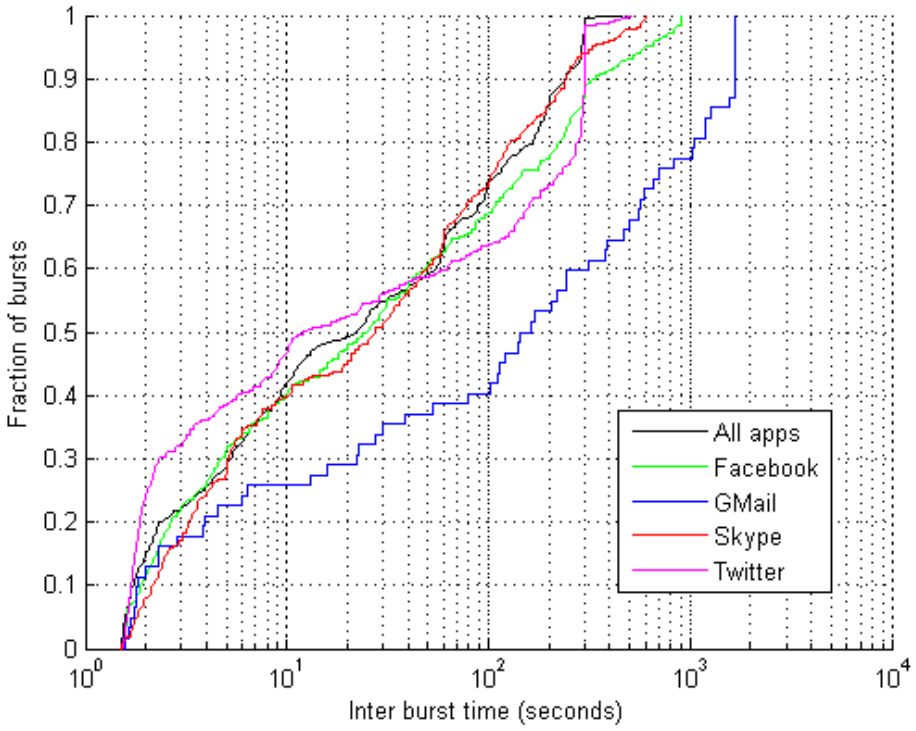
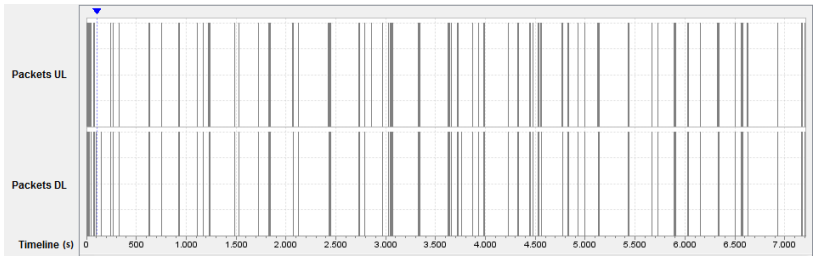
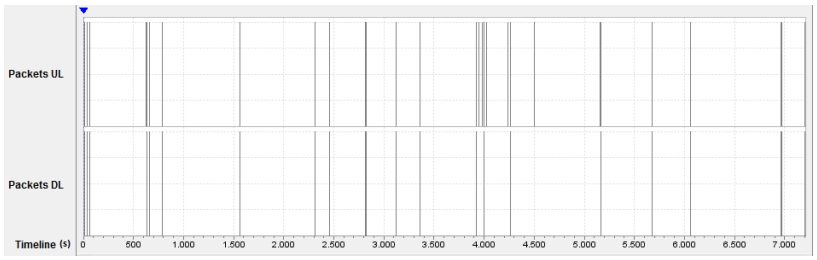


Figure 3.2: Cumulative distribution of inter-burst times for each application

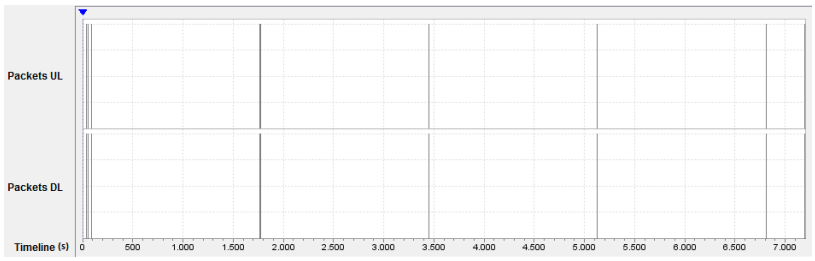
Using the data analyzer tool of the ARO, we also obtain images that visualize network activity in each trace. Figure 3.3 shows samples of traces from each application. Each image in the figure includes vertical grey lines representing uplink (shown as "Packets UL") and downlink (shown as "Packets DL") packet transmissions. We can see the consistence between these images, and the results we obtained in Table 3.1 and Figure 3.2.



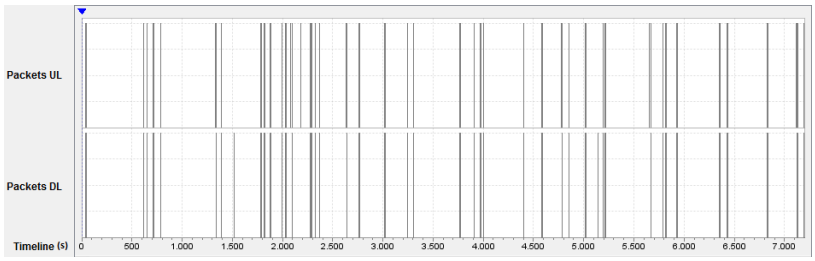
(a) All apps



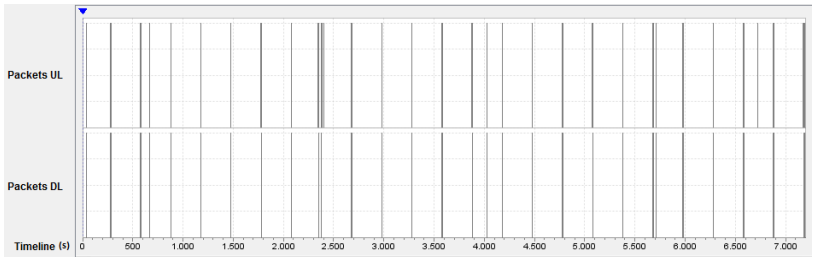
(b) Facebook



(c) GMail



(d) Skype



(e) Twitter

Figure 3.3: Uplink and downlink packet traffic of sample data collection sessions

3.2.3 Pull Based vs. Push Based Transmissions

Mobile applications can be divided into two categories in terms of starting data transmission. The first group of applications are "push based", where the transmission is initiated by a publisher, or a central server. The second group of applications are "pull based", where the application itself initiates the transmission by requesting. If we look from the mobile device side, push based transmissions start with a downlink packet, and pull based transmissions start with an uplink packet.

Since the packet delaying method focuses on reducing the number of radio wake-up events by delaying uplink packets, we can assume that the method achieves higher performance for pull based applications. Because, pull based applications start each transmission request by sending an uplink packet. So, the packet delaying method is able to defer requests starting from their initial points.

In this analysis, we study the distribution of uplink and downlink packets that awaken the 3G modem for all traces, and find the degree of applications' being pull based.

In order to find the points in data traces where the 3G modem is awakened, we again check the packet inter-arrival times. We assume the 3G modem is switched into the sleeping (IDLE) state in any inter-arrival time longer than α seconds. We define α as $t_1 + t_2$, where t_1 denotes the inactivity timer for Cell_DCH→Cell_FACH, and t_2 denotes the inactivity timer for Cell_FACH→IDLE demotion, as we discussed in 2.1. Thus, we guarantee that α is sufficiently long for switching the 3G modem to the sleeping state.

Using a power meter, Qian et al. [17] measured changes in the power consumption of 3G modem in a mobile device during transmission of a single packet. They found the timeout value of the t_1 timer as 5 seconds and the t_2 timer as 12 seconds. Therefore, we set the α as 17 seconds.

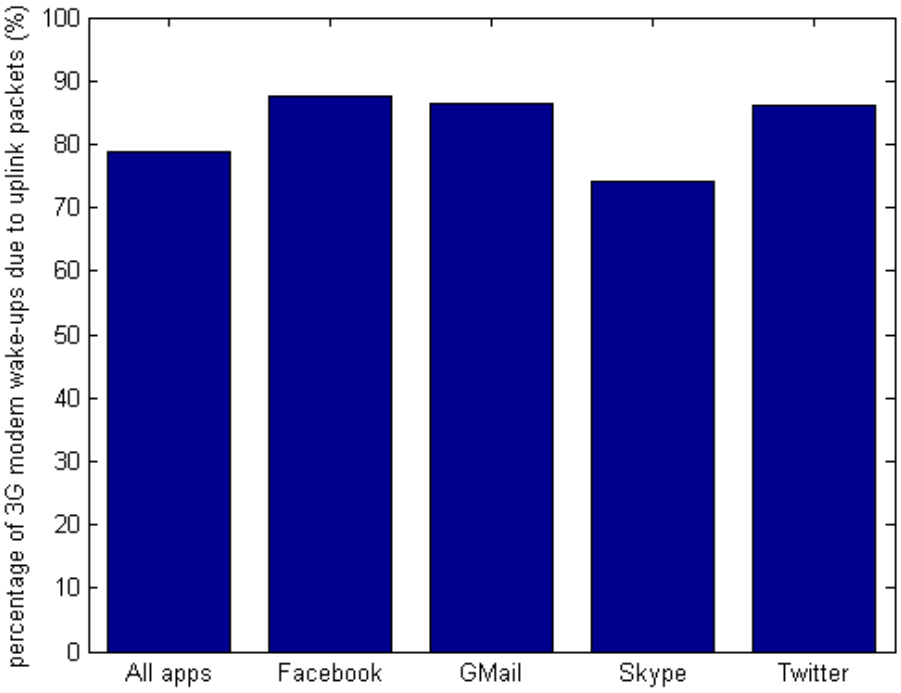


Figure 3.4: Percentage 3G modem wake-up events due to uplink packets for each individual application, and all applications combined

Figure 3.4 shows that, for all applications in our test set, 3G modem wake-up events due to uplink packets dominate downlink packets. Thus, these applications are mostly pull based. This result shows the potential for the efficiency that the packet delaying method can achieve.

3.2.4 Transport Layer Analysis

The transport layer provides logical end-to-end communication between application processes running on different hosts. It is also responsible for maintaining quality and reliability of the communication. Transport layer analysis plays an important role on understanding network packet traffic patterns. Thus, in this section, we analyse transport layer characteristics of our packet traces.

We first calculate percentages of two well-known transport layer protocols: TCP and UDP. For all traces we collect from each application, we obtain results shown in Table 3.2. Being consistent with previous large-scale measurement studies [11, 7], we find that TCP packets constitute the majority of packet traffic for all traces. Therefore, we direct our focus on TCP traffic.

Table 3.2: Percentages of TCP and UDP packets

Application	TCP packet percentage	UDP packet percentage
All applications	93.7	6.3
Facebook	95.3	4.7
GMail	94.9	5.1
Skype	87.8	12.2
Twitter	97.0	3.0

Similar to the TCP analysis made in a previous study [16], we divide the TCP traffic in our traces into four categories for better understanding: (i) connection

control, (ii) data transfer, (iii) loss and recovery, and (iv) other.

We assign normal ACK packets, and packets containing SYN, FIN and RST flags to the connection control category. We look for TCP packets with data payload for the data transfer category. We associate duplicate ACK and data packets with loss and recovery category. Finally, we leave the remaining TCP packets to the other category.

After we calculate the percentages of our TCP analysis categories over total TCP traffic in all traces, we obtain the results shown in Table 3.3.

Table 3.3: Percentages of TCP traffic analysis categories

Application	Conn. Control	Data Transfer	Loss & Recovery	Other
All applications	51.9%	18.3%	2.5%	27.3%
Facebook	56.3%	10%	4%	29.7%
GMail	56.3%	15.5%	5.1%	23.1%
Skype	56.6%	17.1%	1.7%	24.6%
Twitter	50.2%	14.3%	1.9%	33.6%

As we see from the table, actual data transfers contributes to only a small portion of overall TCP traffic. On the other hand, connection control related TCP packets constitute more than half of the TCP traffic for all traces. This finding shows an agreement with our packet size analysis in §3.2.1. Even though connection control packets have no payload, they contribute to more than 50% of TCP traffic.

Chapter 4

PACKET DELAYING METHOD

In this chapter, we introduce the packet delaying method. We provide the main purpose and other design decisions in the system design section. In the same section, we also describe the architecture of our method. In the following section, we give a detailed explanation of the packet delaying algorithm. In the last section, we discuss implementation details of our method.

4.1 System Design

The main goal of the packet delaying method is to reduce the energy spent by the 3G modem when the mobile device is in the background mode. In order to reduce the energy consumption, the method decreases the number of wake-up events of the 3G modem, by delaying outgoing network packets for a predefined period, and sending them together.

While trying to achieve the energy efficiency, we also take the following concerns into account:

1. Preserving the user experience

2. Being able to run with existing applications without requiring any modification

to the application code

3. Not increasing the packet traffic density

To comply with the first item, we design the packet delaying method to operate only during the background mode. The foreground traffic can include delay intolerant packets generated by user interactive tasks such as web browsing, video streaming, or voice-over-IP calling. On the other hand, the background network traffic is mostly delay tolerant [14]. Thus, we can assume that effects of the method on user experience is negligible. For the second item, the packet delaying method does all the packet activity monitoring and delaying operations below the application layer. So, it requires no modification to application code. For the last item, we designed the packet delaying method so that it eliminates uplink packet retransmissions due to long delay periods. Thus, we prevent increasing the packet traffic density with retransmitted packets.

Figure 4.1 shows the system design of the packet delaying method. The system includes two components: a C program, called netfilter queue, that runs at the native process layer, and a control module that runs at the application layer of the Android platform. The netfilter queue program intercepts all the uplink (outgoing) and downlink (incoming) network packets, and implements a buffer to delay uplink packets. It also sends a notification message to the control module whenever it intercepts a packet. The control module implements the packet delaying algorithm, and notifies the netfilter queue to start and end packet delaying sessions.

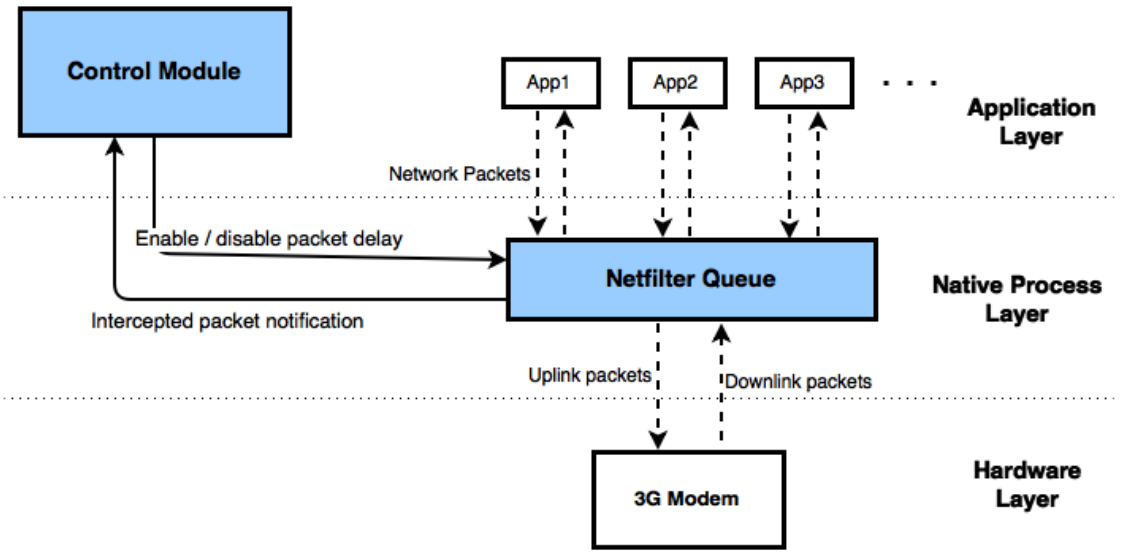


Figure 4.1: System design of the packet delaying method

4.2 Packet Delaying Algorithm

The packet delaying algorithm, as seen from Algorithm 1, includes two modes: the packet delaying mode and the default mode. During the packet delaying mode, uplink packets are routed into a packet buffer before sending them. The algorithm keeps these packets in the buffer until the packet delaying mode ends. When it ends, starting from the first packet that is put into the buffer, all of the packets are sent in order. During the default mode, network packets are directly sent without any delaying.

Algorithm 1 Packet delaying algorithm

Inputs: T_{delay} {Packet delaying period}

C_{buffer} {Maximum number of packets that delaying buffer can hold}

State Abbreviations: DM {Default mode}

PDM {Packet delaying mode}

```
1: CurrentMode ← DM
2: while true do
3:   if Screen is OFF and Cellular modem in idle state then
4:     CurrentMode ← PDM
5:     Start the timer
6:   end if
7:   if CurrentMode == PDM then
8:     if Uplink packet arrives then
9:       Add the packet into the buffer
10:      Delete previously added identical packets
11:    end if
12:    if Buffer capacity == C_buffer or Downlink packet arrives or Timer >= T_delay
       or Screen is ON then
13:      CurrentMode ← DM
14:      Send all packets in the buffer
15:      Reset the timer
16:    end if
17:  end if
18: end while
```

The following conditions are necessary to start the packet delaying mode: The screen of the mobile device should be off, and the cellular modem should be in the idle mode. The first condition guarantees that the user is not directly interacting with the mobile device. In other words, the device is in the background mode. If the cellular modem is awake, then directly sending packets is more effective than delaying them. Because, as the modem is awake, it is already in the high power consumption state. This is why the second condition is also checked before switching to the packet delaying mode.

The packet delaying algorithm switches from the packet delaying mode to the default mode after any of these events: Arrival of a downlink packet, opening of the device screen, ending of the packet delaying period and filling of the packet buffer up to its capacity. Then, the algorithm sends all of the delayed packets in the packet buffer, starting from the oldest one.

4.2.1 Excluding Identical Packets from the Buffer

During the packet delaying periods, we observe that some packets are buffered multiple times. Therefore, these packets are sent multiple times when the packet delaying period ends. This issue cause an overhead in the number of packets in the packet traffic. To prevent the overhead, we check if there are identical packets in the buffer, whenever a new packet is buffered. To check for equality, we compare packets' transport and internet protocol headers. We delete all of the previously buffered identical packets. We observe a significant reduction in the density of the packet

traffic after we test this implementation.

4.3 Implementation

We implement the packet delaying method on the Android Operating System. We run the method on the same test device described in §3.1.1.

We implement the control module, introduced in §4.1, as an Android application. The application includes an Android service that enables running the packet delaying algorithm in the background. Upon starting, the service class also starts the netfilter queue program, also introduced in §4.1, on a separate process. Inter-process communication (IPC) between the control module and the netfilter queue program is handled through read/write operations on a pipe file.

The netfilter queue program implements a buffer to delay uplink packets. To access intercepted uplink and downlink packets, it uses the open source libnetfilter queue library [12]. The library provides an API to packets queued by the kernel packet filter.

In order to route downlink and uplink network packets to the netfilter queue, we use the iptables program. The iptables is a user space program that enables handling tables provided by the Linux kernel firewall and manipulating the network traffic. Before running the netfilter queue program, the control module runs two iptables commands that are seen in Listing 4.1. These commands add rules for downlink and uplink network packets to be directed to the netfilter queue program. Downlink packets are directly forwarded by the netfilter queue program without buffering. The

sole purpose of routing them is to detect arrival of a downlink packet.

Listing 4.1: Iptables rules

```
iptables -A INPUT  -j NFQUEUE
iptables -A OUTPUT -j NFQUEUE
```

To get battery percentage, network connectivity and screen state information, the control module uses the API provided by the Android framework. The control module also calculates the general and packet delaying method's CPU utilization overhead by reading the CPU related time values from the virtual /proc directory of the Linux file system.

Chapter 5

EVALUATION

In this chapter, we evaluate performance of the packet delaying method. We first introduce the experimental setup. In the following sections, we show results of testing the packet delaying method with different delay periods, and effects of applying the method on different application categories. We finalize the chapter by showing the CPU overhead of running the method.

5.1 Experimental Setup

We run the experiments on the Motorola Razr-i smartphone, which we described its properties in detail in §3.1.1. To study effects of the packet delaying method on the packet traffic, we should collect TCPDUMP traces. Thus, during experiments, we also run the ARO data collector application, which includes a TCPDUMP binary.

We collect all the packet traces at various times of the day between 8 A.M. and 10 P.M., in a fixed indoor location. During experiments, we also keep the device screen off. We perform the experiments in AVEA's 3G network that implements HSPA+ technology. Before each data collection session, we stop all running third party mobile applications, except for the ones in our test set we introduce in the following section.

5.1.1 Application Set

In our experiments testing different delay periods, we use the same application set we introduced in §3.1.2. For experiments testing different application categories, we require more applications to represent different categories. Thus, we add the following well-known applications to our application set. These applications also generate background network traffic.

BBC News: A news reader application that includes push notification settings. We enable push notifications and allow 3G usage for downloading contents.

CNN News: A news reader application that includes a background process for fetching breaking news. We allow 3G usage for downloading contents.

WeChat: An instant messaging application. Apart from its application process, it runs a background process to notify users for an incoming message.

WhatsApp: A well-known voice over IP and instant messaging application. It runs a background service on its application process that checks for an incoming message.

YahooMail: An e-mail service application provided by Yahoo. We use one e-mail account. Similar to GMail, this application runs a background process checking for new e-mails.

5.1.2 Trace Driven Simulation

To calculate the total amount of 3G modem related energy consumption during each trace, we apply the trace driven simulation method introduced in [16] by using

the ARO data analyser tool. The method runs a RRC state machine on a TCPDUMP trace, and infers total time spent in each RRC state. Using this inference, it estimates total amount of energy spent by 3G modem. The method is validated in the same study by using a power meter, and found to have 95% accuracy on average.

We use the RRC state machine described in §2.1 for the simulation. For the state machine parameters, we use the parameters inferred in [17] for 3G network called "Carrier 1". The parameters are listed in Table 5.1. The amount of power consumption during IDLE state is subtracted from other power consumption parameters in the table. In reality, the IDLE state includes some CPU and negligible amount of 3G modem activity [21], and has a relatively small amount of power consumption.

The parameters we use may differ from the actual RRC state machine parameters implemented by AVEA, and hence the estimation accuracy can go below 95%. However, our evaluation is not affected as we make a comparative analysis of energy consumption values in different cases.

5.2 Comparison of Different Packet Delay Periods

In this experiment, we compare effects of different packet delay periods on energy consumption and packet traffic. Together with the packet delay method and the ARO data collector, we run Facebook, GMail, Twitter and Skype applications. We collect 9 traces with 2-hour duration for each one of 1, 3, 5, 7 and 9 minute-long packet delay period settings, which makes 45 traces in total. We empirically pick these relatively long packet delay periods to see the extent of delay tolerance in background network

Table 5.1: RRC state machine parameters for trace driven simulation

Parameter	Value
Power cons. in Cell_DCH	800 mW
Power cons. in Cell_FACH	460 mW
Power cons. in IDLE	0 mW
IDLE→Cell_DCH power cons.	550 mW
IDLE→Cell_DCH power cons.	700 mW
Cell_DCH→Cell_FACH timer (t1)	5 sec.
Cell_FACH→IDLE timer (t2)	12 sec.
IDLE→Cell_DCH promotion delay	2 sec.
Cell_FACH→Cell_DCH promotion delay	1.5 sec.
Cell_FACH→Cell_DCH buffer threshold	Uplink: 543 ∓ 25 bytes Downlink: 475 ∓ 23 bytes

traffic. To use as a basis for comparison, we also collect 9 additional traces without running the packet delaying method.

To see the degree of packet delaying activity for each delay period setting, we calculate percentages of total time spent in delay periods, and total number of delayed packets in all traces. As seen in Figure 5.1 (a), time spent during packet delay periods constitutes at least 80% of total trace duration for each delay period setting. The same figure (b) shows percentage of the number of delayed uplink packets. Except for

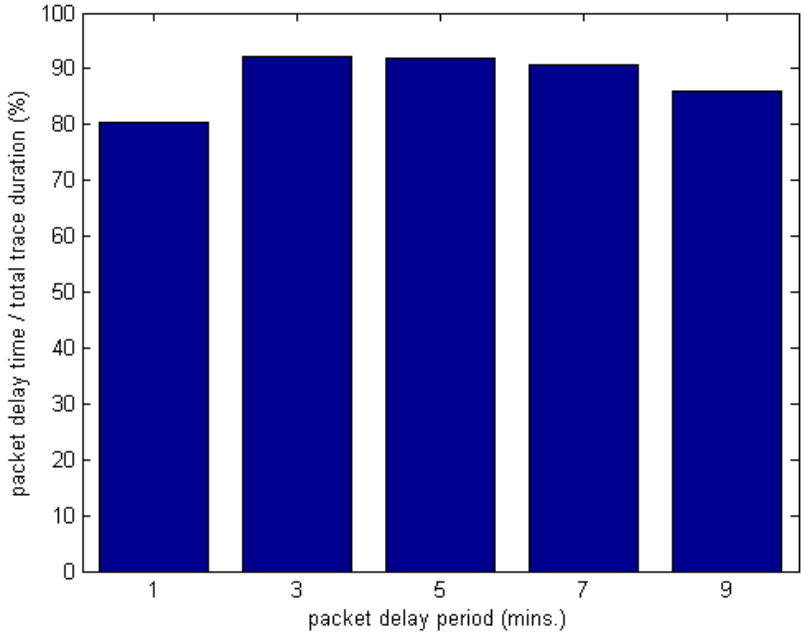
the 1 min. setting, percentages of delayed packets vary between 60% and 70%. For the 1 min. setting, we observe slightly less than 50% of uplink packets are delayed.

We make trace driven simulation on all traces and derive energy consumption values. For each delay period setting and for traces not involving packet delaying, we obtain mean and median energy consumption values listed in Table 5.2. Moreover, we also generate cumulative distribution of energy consumption values in each trace for different delay periods. We obtain results shown in Figure 5.2 (a). As seen from the figure, applying longer delay periods result in less energy consumption. Only 1 minute delay setting incurs higher energy consumption value than no delay setting.

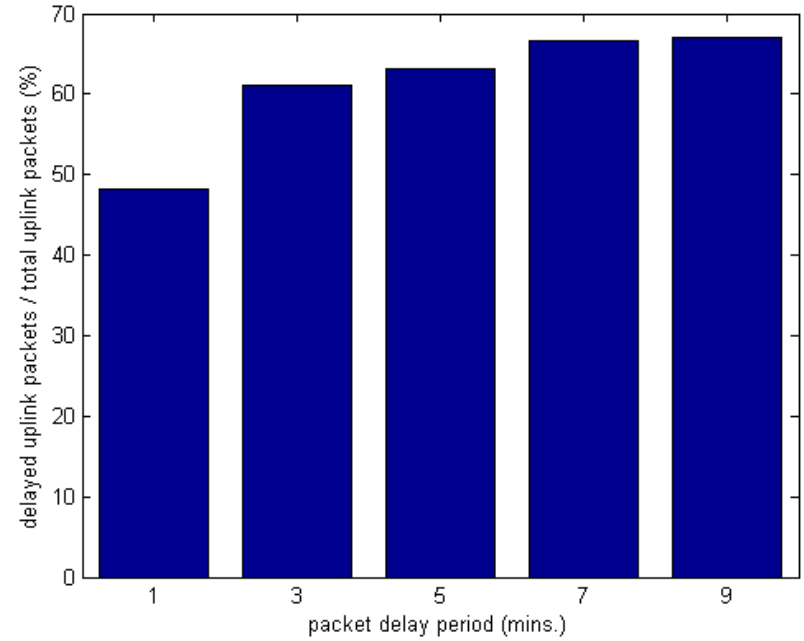
Table 5.2: Mean and median amounts of energy consumption for different packet delay periods

Packet delay period	Mean (joule)	Median (joule)
1 min.	750	710
3 mins.	660	540
5 mins.	710	610
7 mins.	530	540
9 mins.	470	440
No delay	750	710

To see effects of applying different delay periods on the number of packets, we generate cumulative distribution of the number of packets in each trace for different

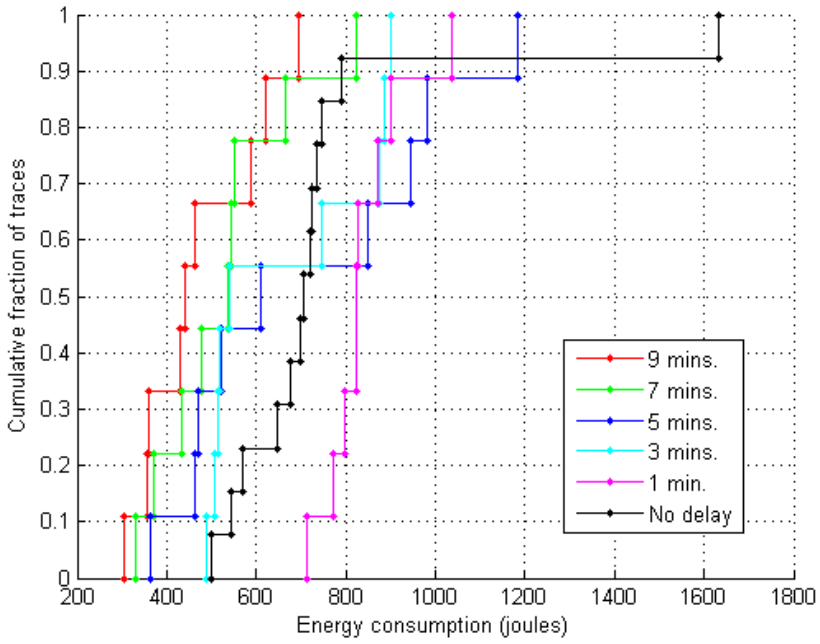


(a)

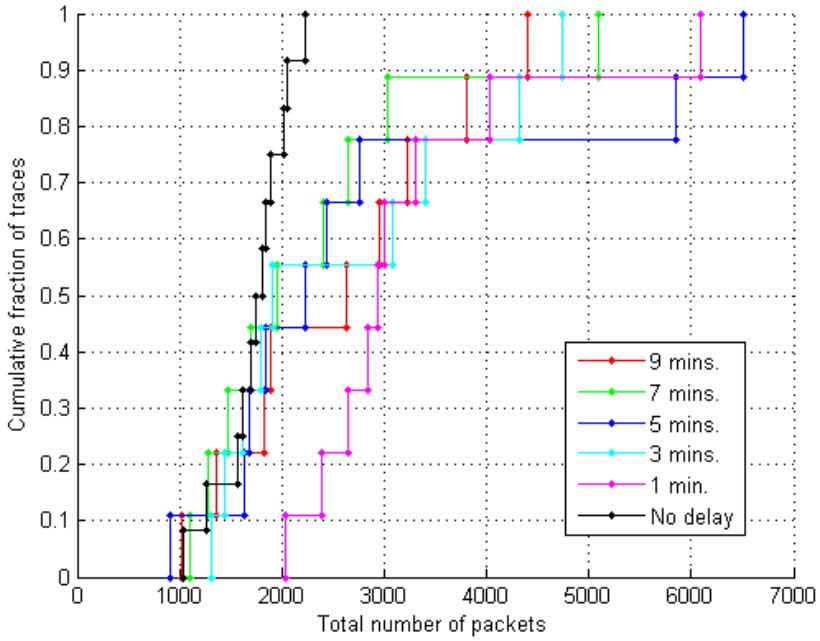


(b)

Figure 5.1: Percentages of (a) total packet delay time over total trace durations, and (b) delayed packets over total number of uplink packets for each delay period



(a)



(b)

Figure 5.2: Cumulative distributions of (a) energy consumption and (b) number of packets in all traces for different packet delay periods

delay periods. We obtain results shown in Figure 5.2 (b). For all delay periods except for 1 minute setting, we observe similar amount of increase in the number of packets. For 1 minute setting, the increase is more profound.

As we discussed earlier, short packet traffic bursts with small amount of data transfer is a sign of inefficient 3G modem utilization. To investigate effects of different packet delay period settings on the number of bursts, we make a burst calculation. For this analysis, we use the burst definition introduced in §3.2.2. Figure 5.3 shows the reduction in the number of bursts relative to those in traces without the packet delaying method applied. As seen in the figure, except the 1 minute of delay setting, we observe up to 40% reduction in the number of bursts.

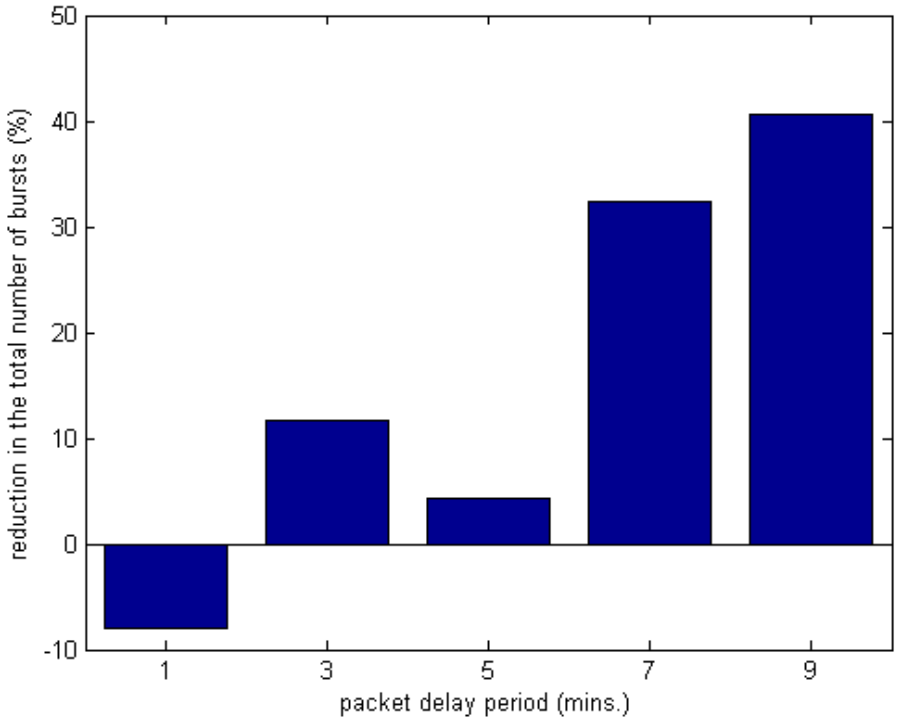


Figure 5.3: Reduction in the number of bursts for different packet delay periods

To investigate impact of different delay periods on TCP traffic, we make the same analysis we did in §3.2.4. We obtain percentages of different TCP packet categories over total number of TCP packets, which are listed in Table 5.3. As we see from the table, the percentage of packets related to TCP connection management increases with the length of packet delay periods. On the other hand, percentages of TCP packets with data, and TCP packets related to packet loss and recovery do not show a significant change.

Table 5.3: Percentages of different TCP packet categories over total number of TCP packets, for different delay settings

Delay Setting	Conn. Management	Data Transfer	Loss & Recovery	Other
No delay	51.9%	18.3%	2.5%	27.3%
1 min.	54.3%	15.9%	4%	25.8%
3 mins.	55.3%	17.8%	3.3%	23.5%
5 mins.	58.2%	18.6%	3.1%	20.1%
7 mins.	58.6%	17.1%	3.0%	21.4%
9 mins.	60.2%	18.0%	3.2%	18.6%

5.3 Impacts of the Packet Delaying Method on Different Application Categories

In this experiment, we evaluate performance of the packet delaying method on different types of applications.

We select four different application categories for the experiment: Instant Messaging, Mail, News and Social Media. We use two applications per category. These applications are listed in Table 5.4. For each application category, we collect 12 traces with 1 hour duration. During 9 of these traces, we apply 3 minutes of packet delaying. We do not apply packet delaying during other 3 traces to use them as a basis for comparison.

Using the trace driven simulation discussed in §5.1.2, we generate energy consumption values for each trace, and calculate the percentage of energy savings for each application category over traces without the packet delaying method applied. Figure 5.4 shows that, the packet delaying method achieves considerable energy savings except for the instant messaging category. Moreover, using the burst definition we introduced in §3.2.2, we analyse the change in the total number of bursts for each category, and obtain results in Figure 5.5. Similar to energy savings results, we observe a significant reduction in the number of bursts except for the instant messaging category.

An explanation for the energy inefficiency we observe in the instant messaging category is the significant increase in the UDP packet traffic. In traces we apply packet delaying to instant messaging applications, we observe around 8-fold increase

Table 5.4: Application categories and corresponding applications

Application Category	Applications
Instant Messaging	WeChat, WhatsApp
Mail	GMail, YahooMail
News	CNN News, BBC News
Social Media	Facebook, Twitter

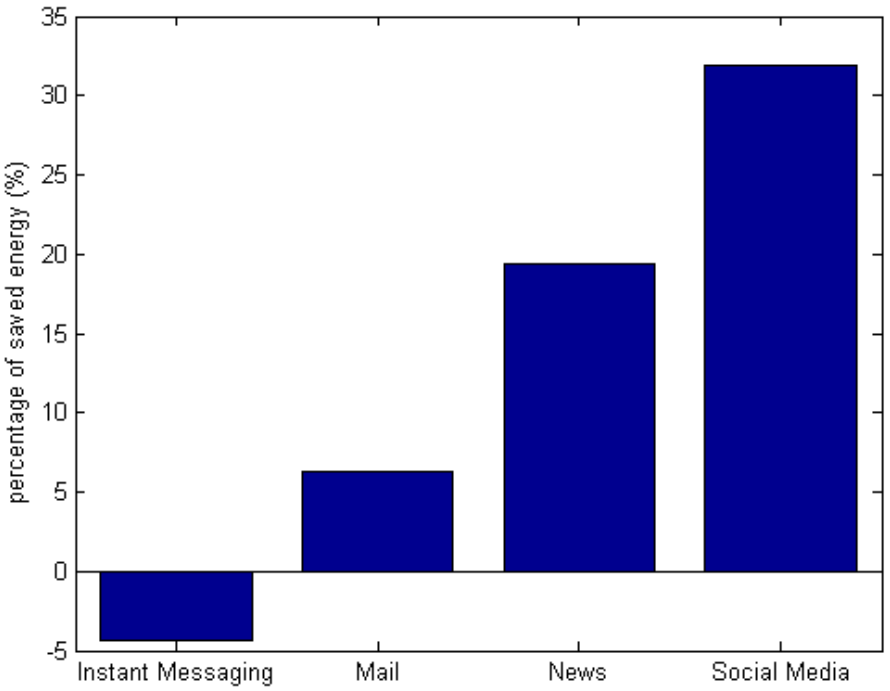


Figure 5.4: Percentage of energy savings for different application categories

in the number of UDP packets compared to traces without packet delaying. The increase in the number of UDP packets also raises the number of packet bursts, and

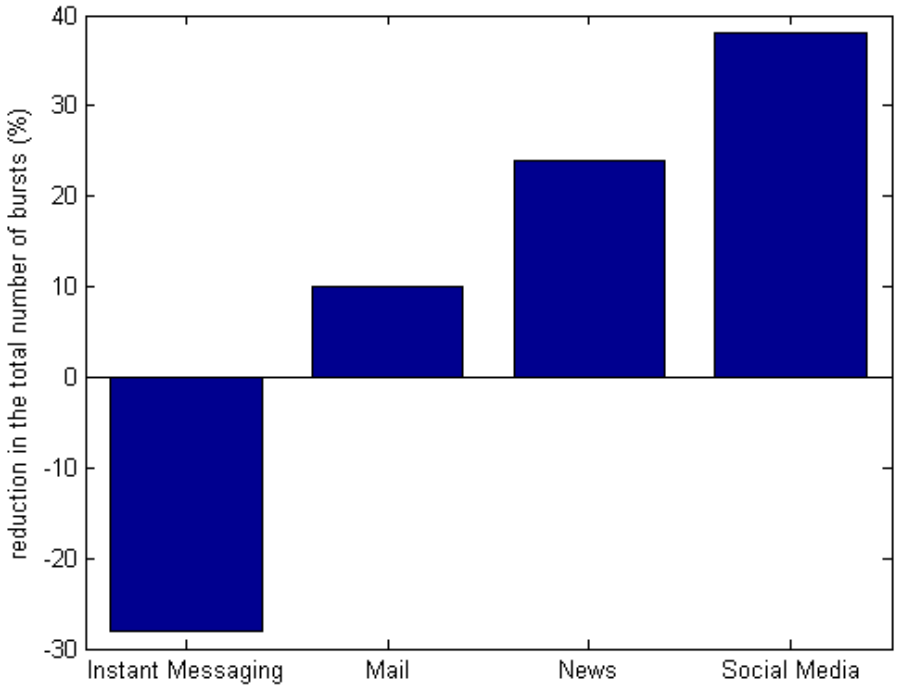


Figure 5.5: Reduction in the number of bursts for different application categories

hence the duration 3G modem spent in high power consuming state.

5.4 CPU Overhead of Running the Packet Delaying Method

Since the main goal of the packet delaying method is to reduce energy consumption, the method itself should also be lightweight. In this section, we analyse how much CPU overhead does the packet delaying method incur.

To determine CPU overhead, before and after each test run, the control module of the packet delaying method fetches the total time spent in cpu and idle states from the virtual \proc folder of the Android file system. At the end of each test session,

the control module calculates the CPU utilization percentage according to these time values, and logs them into a text file.

We generate cumulative distribution of the CPU utilization percentage of the packet delaying method for all traces. We obtain the result shown in Figure 5.6. As the figure shows, for all traces, the packet delaying method utilizes less than 0.45% of CPU.

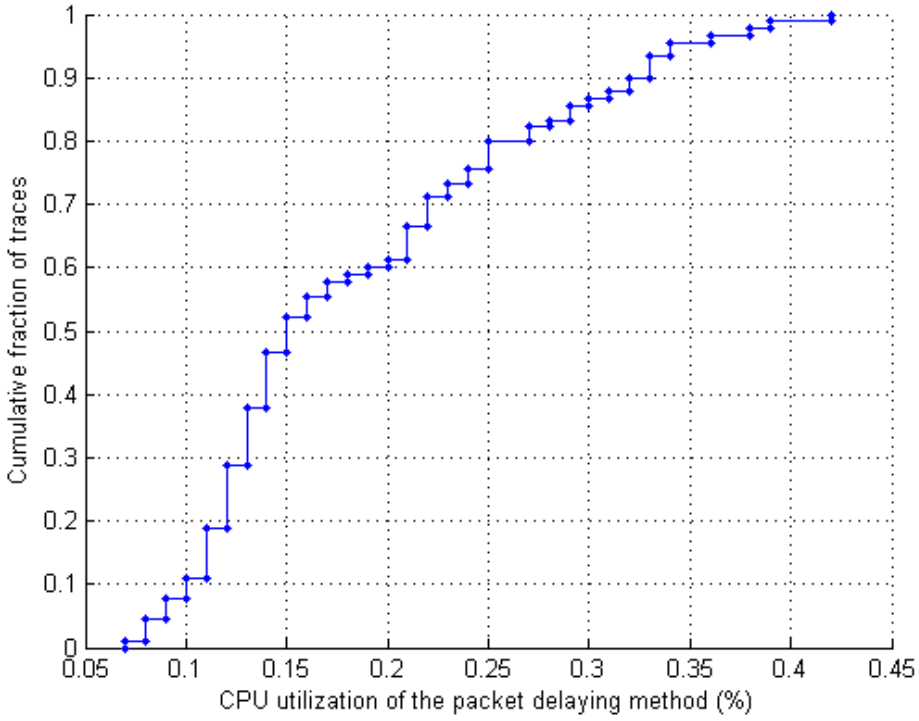


Figure 5.6: Cumulative distribution of the CPU utilization percentage of the packet delaying method for all traces

Chapter 6

CONCLUSION

3G modems are one of the most energy consuming entities in mobile devices. In order to reduce 3G modem related energy consumption, we introduced a traffic-aware packet delaying method. Our method shaped network traffic that is generated by mobile applications running in the background mode. It intercepted and buffered uplink packets for a predefined time period, and sent them together in batches, and reduced the number of times the 3G modem is awakened. We tested the implementation of our method on a Motorola Razr-i smartphone in 3G network of AVEA. We collected background network traffic traces of well-known applications. Using trace driven simulations, we found the packet delaying method can save up to 39% energy for various delay periods and application categories.

In future, performance of the packet delaying method can be further improved by making it application traffic aware. Therefore, the method can apply different packet delaying policies to different application categories according to their delay tolerance. Another possibility is to devise an online learning mechanism to adjust the delay period on the run, according to the packet traffic related features. By this way, the packet delaying method can adjust itself in case of changes in packet traffic patterns.

BIBLIOGRAPHY

[1] Pavan K Athivarapu, Ranjita Bhagwan, Saikat Guha, Vishnu Navda, Ramachandran Ramjee, Dushyant Arora, Venkat N Padmanabhan, and George Varghese. Radiojockey: mining program execution to optimize cellular radio usage. In *Proceedings of the 18th annual international conference on Mobile computing and networking*, pages 101–112. ACM, 2012.

[2] AT&T. Application resource optimizer (aro). `https://developer.att.com/application-resource-optimizer`. Online; accessed 17-August-2015.

[3] Niranjan Balasubramanian, Aruna Balasubramanian, and Arun Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *Proceedings of the 9th ACM SIGCOMM conference on Internet measurement*, pages 280–293. ACM, 2009.

[4] Yong Cui, Shihan Xiao, Xin Wang, Minming Li, Hongyi Wang, and Zeqi Lai. Performance-aware energy optimization on mobile devices in cellular network. In *INFOCOM, 2014 Proceedings IEEE*, pages 1123–1131. IEEE, 2014.

[5] 3GPP Design and Discussion Notes R2-075251. 3gpp release 7: Ue fast dormancy behavior. `http://www.3gpp.org/ftp/tsg_ran/WG2_RL2/TSGR2_60/Docs/R2-075251.zip`, 2007. Online; accessed 12-July-2015.

[6] Hossein Falaki, Dimitrios Lymberopoulos, Ratul Mahajan, Srikanth Kandula, and Deborah Estrin. A first look at traffic on smartphones. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 281–287. ACM, 2010.

[7] Aaron Gember, Ashok Anand, and Aditya Akella. A comparative study of handheld and non-handheld traffic in campus wi-fi networks. In *Passive and Active Measurement*, pages 173–183. Springer, 2011.

[8] Heikki Kaaranen, Ari Ahtiainen, Lauri Laitinen, Siamäk Naghian, and Valtteri Niemi. *UMTS Networks: Architecture, Mobility and Services*. John Wiley & Sons, 2005.

[9] H Andrés Lagar-Cavilla, Kaustubh Joshi, Alexander Varshavsky, Jeffrey Bickford, and Darwin Parra. Traffic backfilling: subsidizing lunch for delay-tolerant applications in umts networks. *ACM SIGOPS Operating Systems Review*, 45(3):77–81, 2012.

[10] Hao Liu, Yaoxue Zhang, and Yuezhi Zhou. Tailtheft: leveraging the wasted time for saving energy in cellular communications. In *Proceedings of the sixth international workshop on MobiArch*, pages 31–36. ACM, 2011.

[11] Gregor Maier, Fabian Schneider, and Anja Feldmann. A first look at mobile hand-held device traffic. In *Passive and Active Measurement*, pages 161–170. Springer, 2010.

[12] netfilter.org. The netfilter.org "libnetfilter queue" project. `http://netfilter.org/projects/libnetfilter_queue/`, 2012. Online; accessed 17-August-2015.

[13] Abhinav Pathak, Y Charlie Hu, and Ming Zhang. Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof. In *Proceedings of the 7th ACM european conference on Computer Systems*, pages 29–42. ACM, 2012.

[14] Feng Qian, Zhaoguang Wang, Yudong Gao, Junxian Huang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Periodic transfers in mobile applications: network-wide origin, impact, and optimization. In *Proceedings of the 21st international conference on World Wide Web*, pages 51–60. ACM, 2012.

[15] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Z Morley Mao, Subhabrata Sen, and Oliver Spatscheck. Top: Tail optimization protocol for cellular radio resource allocation. In *Network Protocols (ICNP), 2010 18th IEEE International Conference on*, pages 285–294. IEEE, 2010.

[16] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Mao, Subhabrata Sen, and Oliver Spatscheck. Profiling resource usage for mobile applications: a cross-layer approach. In *Proceedings of the 9th international conference on Mobile systems, applications, and services*, pages 321–334. ACM, 2011.

[17] Feng Qian, Zhaoguang Wang, Alexandre Gerber, Zhuoqing Morley Mao, Sub-

habrata Sen, and Oliver Spatscheck. Characterizing radio resource allocation for 3g networks. In *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, pages 137–150. ACM, 2010.

[18] International Telecommunication Union. The World in 2014: ICT Facts and Figures. `http://www.itu.int/en/ITU-D/Statistics/Documents/facts/ICTFactsFigures2014-e.pdf`, 2014. Online; accessed 22-May-2015.

[19] Guangtao Xue, Hongzi Zhu, Zhenxian Hu, Jiadi Yu, Yanmin Zhu, and Gong Zhang. Smartcut: Mitigating 3g radio tail effect on smartphones. *Mobile Computing, IEEE Transactions on*, 14(1):169–179, 2015.

[20] Jui-Hung Yeh, Jyh-Cheng Chen, and Chi-Chen Lee. Comparative analysis of energy-saving techniques in 3gpp and 3gpp2 systems. *Vehicular Technology, IEEE Transactions on*, 58(1):432–448, 2009.

[21] Jui-Hung Yeh, Chi-Chen Lee, and Jyh-Cheng Chen. Performance analysis of energy consumption in 3gpp networks. In *Wireless Telecommunications Symposium, 2004*, pages 67–72. IEEE, 2004.