When Mice Consume Like Elephants: Instant Messaging Applications

Ekhiotz Jon Vergara, Simon Andersson, Simin Nadjm-Tehrani
Department of Computer and Information Science
Linköping University, Sweden
ekhiotz.vergara@liu.se, siman481@student.liu.se, simin.nadjm-tehrani@liu.se

ABSTRACT

A recent surge in the usage of instant messaging (IM) applications on mobile devices has brought the energy efficiency of these applications into focus of attention. Although IM applications are changing the message communication landscape, this work illustrates that the current versions of IM applications differ vastly in energy consumption when using the third generation (3G) cellular communication. This paper shows the interdependency between energy consumption and IM data patterns in this context.

We analyse the user interaction pattern using a IM dataset, consisting of 1043370 messages collected from 51 mobile users. Based on the usage characteristics, we propose a message bundling technique that aggregates consecutive messages over time, reducing the energy consumption with a trade-off against latency. The results show that message bundling can save up to 43% in energy consumption while still maintaining the conversation function. Finally, the energy cost of a common functionality used in IM applications that informs that the user is currently typing a response, so called typing notification, is evaluated showing an energy increase ranging from 40-104%.

Categories and Subject Descriptors

C.2.1 [**Computer Communication Networks**]: Wireless communication; C.4 [**Performance of Systems**]: Measurement techniques

General Terms

Design, Measurement

Keywords

instant messaging; transmission energy; UMTS; mobile devices; typing notification

e-Energy'14, June 11–13, 2014, Cambridge, UK.
ACM 978-1-4503-2819-7/14/06.
http://dx.doi.org/10.1145/2602044.2602054.

1. INTRODUCTION

Instant messaging (IM) applications have emerged as the substitute for Short Message Service (SMS) and have gained wide popularity. These applications offer the possibility of sending text messages (1-to-1 or to a group) as well as other multimedia messages (e.g., images, audio or video). Applications such as WhatsApp or QQ have already 400 and 800 million online users respectively [3, 4], and recently, IM has overtaken the traditional SMS text messages [1]. Given the widespread use of IM, even home appliance manufacturers envision its usage for controlling their equipment [2].

While this might seem a blessing to the user, IM text messages are an example of a type of traffic with low bandwidth requirement, which leads to high energy consumption. The exchange of a couple of text messages can consume as much as sending an image due to the radio resource allocation of cellular networks. From the cellular network operator perspective, the signalling overhead created by IM is very high given their intermittent and small data transmissions.

IM applications provide more functionalities than regular SMS, such as online presence awareness, typing notification or status updates. Application developers may unfortunately integrate these features without studying the potential impact on energy consumption. A recent study analysed more than 9 million comments from the Google Play Store and showed that more than 18% of all commented applications have negative comments regarding energy consumption [28].

The result is that different applications delivering similar function consume completely different amounts of transmission energy. We selected 6 of the most popular IM applications from the Play Store on 15th January 2013 as an illustrative example, and sent the same 2 minutes conversation between two smartphones connected via 3G using the different applications. The energy consumption for each application was computed using EnergyBox [26], our tool that is described briefly in section 6.2.

Fig. 1 (top and bottom-right) shows a great diversity regarding the amount of energy spent and data sent by the different applications when performing the short conversation. The most consuming application (Messenger) consumes 153% more energy than the least consuming one (GTalk) to transmit the same conversation. Fig. 1 (bottom-left) shows a significant diversity in the packet size for the 3 selected applications, which impacts the transmission pattern, and thus the radio resource allocation and energy consumption. For example, WhatsApp employs smaller packets than GTalk, but performs transmissions more often leading

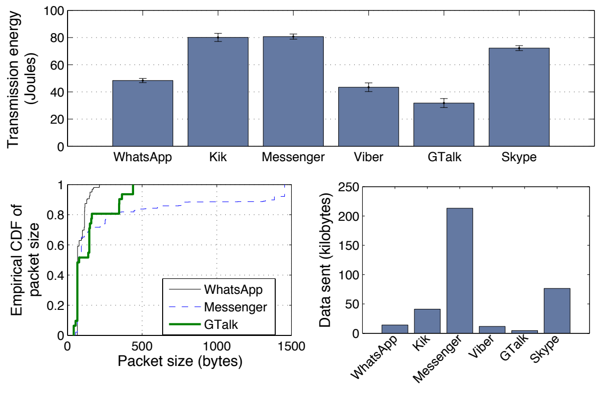

Figure 1: Average transmission energy, amount of data sent and empirical CDF of packet size for different Instant Messaging applications exchanging a short conversation.

to higher energy consumption. Kik, Messenger and Skype transmit more data than the others, making the 3G interface consume more. Using the least energy-efficient application could substantially shorten the battery lifetime of a device, by a factor of 2.5, and reduce the quality of experience (QoE) for the user.

The transmission pattern of IM is mainly determined by user interactions, where interactive traffic is generated by a sequence of exchanged messages. Thus, the complexity of designing energy-efficient transmissions increases given the a priori unpredictability of the users. However, studying current usage patterns can reveal inefficient ways of performing transmissions since neither the users nor the applications are aware of the energy footprint characteristics.

The contributions of our work, which aims to significantly reduce the energy consumption of IM for mobile devices, are threefold:

- We collect, analyse and provide an IM text message dataset[1] of 1043370 messages from 51 mobile users that describes users' diverse usage patterns.
- We demonstrate the high energy cost of a networking functionality, typing notification, that most IM applications implement amounting to additional energy consumption between 40-104%.
- Informed by the usage patterns found in our dataset, a message bundling algorithm is proposed showing that aggregating consecutive messages from the same user saves up to 43% energy.

The paper is organised as follows: section 2 explains the background and describes the related works. Section 3 analyses the IM dataset of real user messages. Section 4 describes the proposed message bundling technique, followed by the algorithm applied to the typing notification feature in section 5. The evaluation methodology is presented in section 6, and section 7 and 8 present the results. Finally, the conclusions and future work are presented in section 9.

[1]http://www.ida.liu.se/~rtslab/energy-efficient-networking

2. BACKGROUND AND RELATED WORKS

We begin by providing an overview on the communication energy footprint for the third generation Universal Mobile Telecommunications System (UMTS) at the user equipment (UE) side. The main related works are presented in section 2.2.

2.1 Energy footprint of 3G

The energy consumption of the UE when connected to a 3G UMTS network is mostly influenced by the radio resource management performed at the network operator side by the Radio Network Controller (RNC). The RNC employs the Radio Resource Control (RRC) and Radio Link Control (RLC) of the UMTS Wideband Code Division Multiple Access protocols to perform the radio resource management of the UE [11].

According to the RRC, the UE implements a state machine where the different states have different power consumption and performance in terms of maximum data rate and latency. The UE states are CELL_DCH or Dedicated Channel (DCH), CELL_FACH or Forward Access Channel (FACH), and URA_PCH or Paging Channel (PCH), sorted from highest to lowest power drain and performance in terms of data rate and response time. Since the states URA_PCH and CELL_PCH result in similar energy consumption, we consider them as PCH for simplicity.

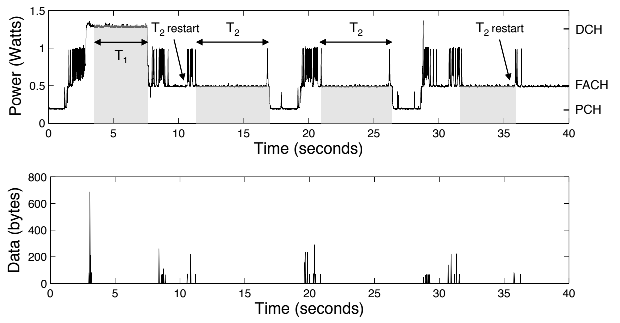

Figure 2: Example power profile for 3G using a mobile broadband module (Ericsson F3307) and Skype as instant messaging application.

Fig. 2 shows the observed power consumption levels of the different states at one location for the state machine implemented by the operator TeliaSonera in Sweden. The bottom graph shows the packets when they were captured at the network interface of the UE. In PCH, the UE cannot transmit any data, but it can be paged with the lowest energy drain. When the UE starts generating or receiving traffic, some signalling is required to establish the connection and move the UE from PCH to DCH before sending any data.

The RNC employs the RLC protocol to evaluate the allocation of resources and control state transitions to higher performance states. The UE reports the observed traffic volume to the RNC, which re-allocates the UE state if the RLC buffer data occupancy of the UE exceeds some fixed RLC thresholds. These thresholds control the PCH-DCH and PCH-FACH state transitions. When the PCH-DCH threshold is exceeded, the UE is moved to DCH (3 s in Fig. 2). In DCH, the UE is allocated a dedicated physical channel (uplink and downlink) providing the highest data rates. The first packet is a downlink transmission, and thus the signalling occurs before the data is received at the UE.

The RRC state machine uses inactivity timers to down switch the UE to lower performance states. The UE is moved to FACH after T_1 when there is small or no data transmission (8 s in Fig. 2). In FACH, the UE is assigned a default common or shared transport channel where only low-speed transmission can be performed. Finally, the transition from FACH to PCH is controlled by the inactivity timer T_2.

Inactivity timers create energy overheads known as energy *tails* since the UE remains in a high energy consuming state while not transmitting anything [6, 25]. Fig. 2 shows that the data pattern (inter-packet interval and packet size) clearly influences the energy consumption driven by the RRC state machine, where even small transfers of data can trigger state transitions to DCH (e.g., in the case of a small chat message) or the restart of the inactivity timers.

To sum up, this illustrates that the RRC state machine and the above mechanisms impact the energy consumption of the UE through the data pattern in a complex way.

2.2 Related works

We categorise the related works into two different groups: IM and energy consumption of cellular data transfers at the user end.

IM: Even though IM has a great popularity and a large number of users, little work has been done to understand its energy consumption at the user end. Most IM related works focus either on users' social interactions [10], security [22] or analysing the IM traffic generated in desktop-oriented machines [13, 15, 30].

While presence updates have been studied in desktop-oriented machines [27,30] and in the mobile context [7,8,17], the cost of the typing notification feature or the impact of consecutive messages has not attracted attention so far.

Cellular communication energy: Several works address the general problem of high energy consumption of cellular communication at the user end. The readers interested in other aspects than transmission energy are referred to the survey by Vallina-Rodriguez et al. [24].

EnergyBox [26] is developed to study the energy consumption of 3G and WiFi transmission energy. ARO is a similar tool that at the time of its publication [21] was not available to other researchers. Pathak et al. [18] propose Eprof, a system-call-based energy profiling tool for smartphones, and show that transfers of advertisements in free applications have a great energy cost.

Periodic transfers and background traffic of mobile applications are known to incur great energy consumption. Some proposals [5, 9] employ the Fast Dormancy (FD) mechanism introduced in the 3GPP Release 8 as a radio resource control technique to move the UE to PCH before the expiration of the inactivity timers. Traffic shaping techniques [6, 12, 14, 16, 20] are common. These shift transmissions over time (e.g., batching background traffic) to minimise the transmission cost. For example, our previous work [25] schedules background data transfers considering the inactivity timers and the RLC data buffers in combination.

While the above works attempt at reducing energy footprint for a generic class or a subset of application flows, this work explores tailor-made solutions for IM using application indicators and the user interaction knowledge obtained from our dataset. We also show, by analysing the cost of added functionalities (typing notification), that developers can rethink the inclusion of the function or providing the option to disable them.

3. COLLECTED IM DATASET

Analysing the way users write text messages using IM applications can reveal current inefficiencies in terms of energy consumption. We are interested in the user input, which is translated to network traffic by the applications. Thus, we study a dataset of user text messages collected at the application layer in this section.

The messages were collected from one of the most widely used IM applications (WhatsApp) during a period between 23rd of January 2011 and 8th of January 2014. Note that the logs from different users have different starting time. We agreed on collecting the usage data with the users after they were created (from past logs), thus the data reflects the normal behaviour of the users. The users employ WhatsApp as their primary IM application.

A parser was developed to obtain the messages from the logs of WhatsApp in each user device. Every text message is represented by its timestamp (UNIX time), the message length in characters, the direction (in/out), the user number, the chat type (single chat or group chat) and the chat number. A *chat* is a sequence of messages exchanged with a user (1-to-1 single chat) or a group of users (group chat) over the duration of collection. By *conversation* we will denote a subsequence of all the messages belonging to one of these chats. The user name and the actual content of the message are obfuscated for privacy reasons. The focus of this work is on text conversations, and therefore we do not consider multimedia messages, which are left for extensions of this work.

The dataset currently contains 1043370 messages collected from 51 users. The age of the users is different: 34 users between 25-30 years, 12 users between 30-35 years, and 5 users above 45 years. Regarding the country, most users are from Spain (33) and Sweden (13), whereas the rest are from Belgium (2), Germany (1), USA (1) and Mexico (1). The messages appear in 1815 conversations with other users (2089 users in total).

While selection of representative subset of all messaging patterns in the world would require a careful analysis, we believe that the current dataset provides an interesting subset since (1) it has a diverse user base, and (2) it shows a great diversity in terms of number of messages sent per day as well as used chat types (single or group chat).

The timestamp and message size are the most interesting values from the transmission pattern perspective. We start by analysing the message distribution according to their origin (in/out) and conversation type.

3.1 Message origin

Single chats are the most common type of conversation (83% against 17% for group chats). The larger fraction of messages is originated from single chats (59% of all the messages), whereas the rest (41%) is from group chats. Table 1 shows the number of messages per chat type and direction. The input messages dominate in the dataset (65%).

We observe that there is a great diversity in the above numbers across the different users of the dataset. Fig. 3

Table 1: Message origin per direction and chat type.

	In	Out	Total
Single chat	29%	30%	59%
Group chat	36%	5%	41%
Total	65%	35%	

shows the message classification per user (the users are sorted by the number of single chat messages).

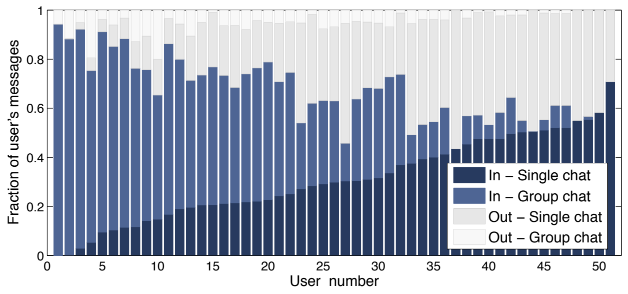

Figure 3: Message classification per user.

Most of the users are shown to have all the message classes, but in different proportion. For example, user 10 has a much larger proportion of group chat input messages compared to user 40 (50% against 5%), meaning that half of the messages received by user 10 are from group chats. However, user 40 has a larger proportion of single chat in and out messages, meaning that the user mostly uses single chats. Some users do not even use group chats (e.g., user 37 or 44).

Regarding the number of chats per user, the average and variance are 35±23 chats. For an arbitrary user, the greatest proportion of messages is typically concentrated in a few chats: On average, 50% of the messages of a user belong to only 2 of her chats (and these are typically single chats). Fig. 4 (left) shows a number of selected users (one curve per each user) according to the message distribution over the chats sorted by message volume in order to illustrate the diversity across the users. The average number of users in group chats is 7. However, larger groups are also present (up to 26 users) but less common. Only 11% of the groups are larger than 12 users.

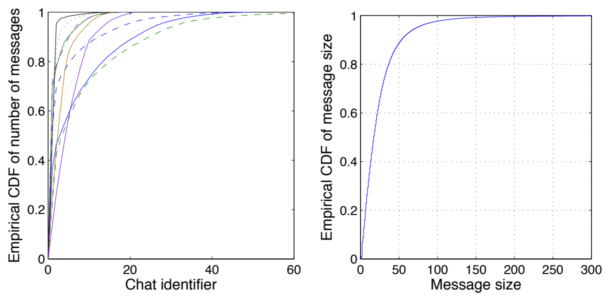

Figure 4: Message distribution of selected users over chat (left) and overall message size distribution (right).

To summarise, the great diversity among users makes the usage depend much on the type of chat they employ. Differentiating these cases is interesting to employ different techniques to save energy. For example, some users might receive many messages from group chats while not using their device, leading to high energy consumption. A downlink message batching policy could reduce the energy consumption for this type of users.

3.2 Message size

Regarding message size, we observe that short messages are predominant. The average message size is 26 characters. Fig. 4 (right) shows that messages shorter than 40 characters comprise 83% of all the messages. Small messages lead to small packets, making it very inefficient to send each message in a separate packet. For example, if the transmission of these small messages is performed over TCP/IP (i.e., the most common case), the overhead created by the packet headers (40 bytes) is the same size as the payload (i.e., the message).

In order to understand which message sizes produce more traffic we multiply the message size by the number of messages. Fig. 5 shows the distribution of the total traffic over the message size and the proportion of the different origins (single/group chat and in/out). The small packets generate most of the traffic.

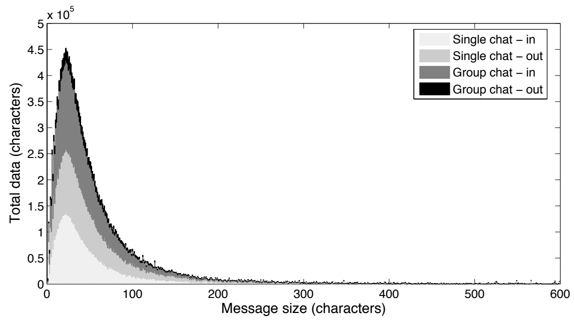

Figure 5: Distribution of the total data sent.

Various works show the benefit of compression for transmitting less network data and thus reducing the energy cost [19, 29]. We analysed the compression performance of the default compression strategy in Android[2] on the actual messages from the collected WhatsApp logs. Since most of the messages are short messages with few recurring patterns, we found that only 2.3% of the messages of our dataset would benefit from compression.

3.3 Temporal properties

Fig. 6 (top-left) shows the distribution of messages appearing in the collected dataset. As expected, the users tend to exchange more messages during the afternoon and the evening. Looking at the aggregates, the number of messages gradually grows during the day and peaks at 20:30 in the evening.

Fig. 6 (bottom) suggests that there is no clear weekly pattern in contrast to the cellular traffic observed in other works [23], where the daily peaks observed on the weekdays

[2]http://www.gzip.org/algorithm.txt

are higher than during weekends. Every day of the week shows a similar trend.

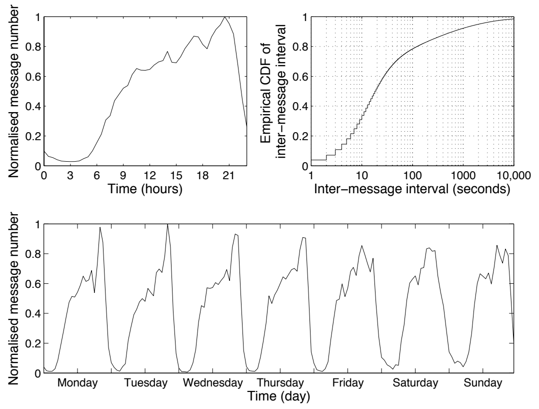

Figure 6: Normalised number of messages over hours in a day (top-left), empirical CDF of inter-message interval (top-right) and normalised number of messages during different days of a week (bottom).

Fig. 7 shows the average and standard deviation of the number of messages sent per day and user. There is a great diversity across the dataset, from very active users (more than 200 messages) to users that exchange few messages. The large standard deviation describes the variation for the same user between days, which makes it difficult to predict the IM traffic only based on the transmitted messages in the past.

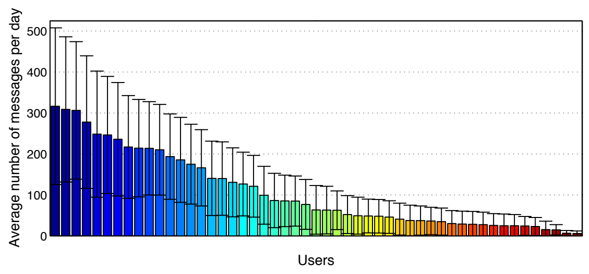

Figure 7: Average and standard deviation of messages per day per user.

The empirical CDF of inter-message interval (IMI) of the dataset is shown in Fig. 6 (top-right). We observe that messages with short IMI are predominant, where 73% of the messages have an IMI shorter than 1 minute. The long tail describes the distinct idle periods of the users.

We have observed that users generally write more than one message in a row before receiving any answer. This type of message is hereafter referred to as *consecutive* message. 48% of the messages in the dataset show this characteristic. From the energy perspective, consecutive messages are wasteful since they restart the inactivity timers of the cellular interfaces leading to higher energy consumption. If the elapsed time is greater than the inactivity timer, the UE consumes another energy tail. The short IMI of most messages, their small size and the significant presence of consecutive messages suggest that these could be coalesced to extend the idle time. In this paper we explore this option and present an algorithm that aggregates them in section 4.

Finally, two types of conversations (i.e., a chunk of the whole chat) can be identified over time. Based on the number of messages sent and their closeness in time, we can distinguish two different types of conversations over time: *sparse* and *dense* conversation periods. Sparse conversation periods have their messages more separated in time (i.e., higher IMI), while the dense ones represent intensive periods of interaction with shorter IMIs. This knowledge is used later on to select the evaluation traces.

To sum up, the dataset provides valuable insights for studying the message exchange characteristics of different users and efficiency of transmissions.

4. BUNDLING OF MESSAGES

Bundling aggregates consecutive messages by sending them together, ideally in the same packet. The potential benefits are the following:

- Less protocol overhead: Since the small messages predominate the IM text traffic, sending the messages in the same packet payload instead of sending them in separate packets would reduce the amount of protocol overhead.
- Extended idle state time: Reducing the number of transmissions by aggregating messages in the same bundle allows the UE to extend its idle state time, and therefore the energy consumption.
- Opportunity for better compression: We conjecture that aggregating messages would improve the compression results. However, the impact of compression is not studied in this paper.

Even though the benefits of message bundling are convincing, the main drawback is that the technique delays messages in order to send them together. Nevertheless, it is unclear whether introducing delay for consecutive messages is detrimental for the QoE when the other user is not active in the conversation.

Algorithm 1 Message Bundling

```
Require: $Q_M$ (initially $\emptyset$), BundleTime
 1: Upon: send button pressed
 2:   m ← text area content
 3:   $Q_M$ ← m
 4:   BundleTimer ← BundleTime
 5: Upon: text input area changed
 6:   BundleTimer ← BundleTime
 7: Upon: BundleTimer = 0
 8: if $Q_M \neq \emptyset$ then
 9:     Transmit $Q_M$
10:     $Q_M \leftarrow \emptyset$
11:     BundleTimer ← BundleTime
12: end if
```

In order to investigate the potential energy savings of message bundling and the impact on QoE in terms of message delay, we propose an event-based algorithm, which uses information from the graphical user interface. Algorithm 1 describes the operation of message bundling. The intuition behind the algorithm is that when the user presses the *send* *button* the message is not directly sent over the network. Instead, a *BundleTimer* is started (or restarted if it is already running). The content of the *text input area* (i.e., the message m) is queued in the message queue Q_M.

Whenever the text input area is changed, it means that the user is typing the next message. Therefore, the BundleTimer is restarted every time the user changes the text, so that the new message can be bundled too. This prevents the Q_M contents to be transmitted while the user is typing, thus recognises consecutive messages.

However, if the user never stops typing, the messages will never be sent. We argue that if the user keeps typing, the messages can be considered as the same one and QoE will not suffer from sending them together, no matter the length of the message. Therefore, we consider this case to be a pathological case.

The BundleTimer is a statically configured countdown timer, with the *BundleTime* as an input parameter. Whenever it expires, the messages in Q_M are transmitted over the network, Q_M is emptied and the BundleTimer is restarted.

The proposed BundleTime parameter provides flexibility: a short BundleTime decreases the sending delay for single messages. A longer BundleTime would allow text over several consecutive messages to be aggregated. Since the "text input area changed" event requires the screen to be switched on, we do not consider more general indicators such as screen for simplicity.

5. TYPING NOTIFICATION

The typing notification is a common feature implemented by most IM applications. The mechanism notifies the (receiving) user when the other (sending) user in the conversation is typing, creating a notion of presence and interactivity. Users can use this information to decide whether they should remain in the conversation.

In order to study the cost of this mechanism, we develop an algorithm, which emulates its operation in a simple IM application. The notify updates should be triggered when the user types a character in the text input area on the graphical user interface. However, it is up to the implementation to decide the frequency of these events resulting in a network packet being sent to the receiving user as a notification. The typing notification feature increases the amount of packets transmitted, and potentially the energy consumption.

Algorithm 2 employs a countdown timer named NotifyTimer to restrict the rate of notify messages. The NotifyTimer is statically configured with a NotifyTime value in seconds. When the user is typing, a text input area changed event is triggered. If the NotifyTimer is not running (i.e., NotifyTimer = 0), the algorithm will send a notification packet to the network and start the NotifyTimer. The next time the user types, the NotifyTimer will avoid a new packet being sent to the network. Thus, if the user is continuously typing the algorithm performs only a single notification transmission every NotifyTime.

When the other user receives the typing notification, the receiving end of the application will show the receiving user the notification in the screen. For example, we observe that WhatsApp uses approximately a 3 seconds timer for the NotifyTime.

Algorithm 2 Typing notification

```
Require: NotifyTimer (initially NotifyTime)
1: Upon: text input area changed
2: if NotifyTimer = 0 then
3:     Transmit notification
4:     NotifyTimer ← NotifyTime
5: end if
```

6. EVALUATION METHODOLOGY

This section describes the methodology and the evaluation environment used to quantify energy consumption of the typing notification and message bundling.

The general methodology is as follows: a predefined set of real conversations is automatically replayed between a pair of clients using a prototype IM application running on commodity devices. The conversations are transformed to network transmissions (i.e., packet traces) by our prototype application. The transmission is performed using the real 3G network described in section 2, and the resulting packet traces are captured. The energy is calculated using EnergyBox from the captured real packet traces.

First, the energy consumption of each conversation is calculated as a baseline (without typing notification nor message bundling). Second, the conversations are replayed with only message bundling enabled (Algorithm 1), and the energy savings are calculated and compared against the baseline. Third, the conversations are replayed with only the typing notification enabled (Algorithm 2), and the extra energy cost is calculated by comparing the resulting energy against the baseline.

First, we briefly describe the evaluation environment as well as the evaluation conversations and EnergyBox.

6.1 Evaluation environment

The client application provides the chat functionality between two Android devices and implements the logic of Algorithms 1 and 2. The implementation of the message bundling and the typing notification feature are based on instances of the Async Task class provided by Android, where the Async Task represents the timers that can be cancelled, reset and started again.

We selected the Message Queueing Telemetry Transport (MQTT) protocol as the transport protocol for the following reasons: it is a lightweight application protocol, some IM applications officially use it (e.g., Facebook Messenger), and the Mosquitto open source project provides easy to set up public servers that accelerate the development phase. MQTT is a publish/subscribe protocol, where the subscribers are instantaneously notified whenever a publisher generates a new event.

Fig. 8 shows the general architecture of the application and the test environment. The IM clients are able to communicate using MQTT through the server that hosts the MQTT broker. Each user subscribes to her nickname. The conversation partners send a message by publishing the message to the nickname (in publish/subscribe terms, the topic) of the recipient. The MQTT broker keeps track of the topic subscriptions.

Since the IM applications tested in section 1 are black boxes, we cannot compare the results obtained from our prototype application against them in a fair way. However, for the interested reader, the basic energy cost of the prototype

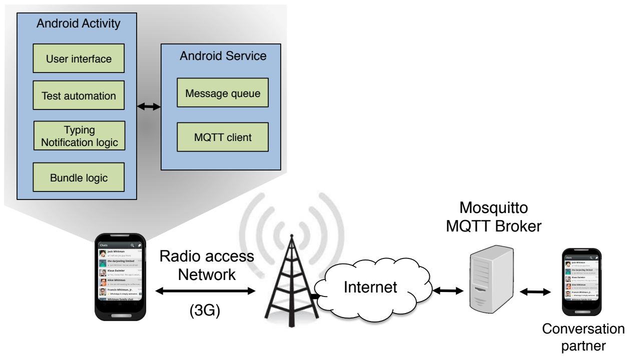

Figure 8: Architecture of the IM prototype implementation and the test environment.

application for the same conversation as shown in section 1 is lower than the rest (18 Joules).

6.2 Test traces and parameter settings

This section describes the parameter settings and the conversations used to study the energy characteristics of the typing notification and the message bundling.

Test conversations: Four different test conversations are selected for the tests based on the different patterns observed in section 3. Two conversations (*Dense* and *Sparse*) were randomly selected from the dataset, representing intensive conversations (short IMIs) and slow-paced ones (longer IMIs). *Random* is a synthetic conversation generated by randomly selecting messages from the dataset not considering the resulting conversation's dense/sparse characteristics. By randomly selecting the messages we aim at obtaining messages of different sizes. The selection follows the following rules: Message pairs in the resulting conversation have an IMI of shorter than 30 s since larger IMIs are uninteresting from the energy perspective (greater than the typical inactivity timers). Moreover, if a selected message appears in a sequence of consecutive messages, then the other messages belonging to that sequence are also selected to keep the realistic consecutive message relations from the dataset. Finally, *Short* is the same conversation used in the introduction to compare the different IM applications. It characterises a trace containing short messages, a single consecutive message, and IMIs between the Sparse and Dense conversations. The four categories have different number of consecutive messages, which is interesting to test the message bundling.

The test conversations used are shown in Fig. 9. The duration of the selected conversations is below 250 s. We believe these are representative conversations for IM. Since our dataset did not delimit conversations (the users were always logged in and no distinction of different conversations were made within the chats), we base our reasoning about duration on earlier work [30], where 9900 of approximately 10000 conversations were shorter than 250 s. Note that in Fig. 9 the conversations do not start from 0 since the time to write the message is also considered, and the difference in their length is not important since the results are studied per conversation.

Writing speed parameters: In order to simulate the user typing and automate the tests, the prototype application is instrumented to automatically write characters at a

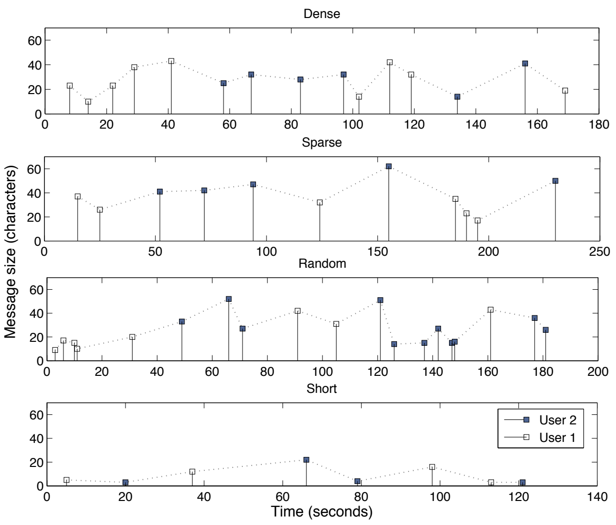

Figure 9: Test conversations.

parametrised writing speed. For each message of a given conversation, the characters of the message are written using a constant typing speed. The automation logic starts writing in the input text field so that the message is sent in its correct timestamp. The time to start writing the message is calculated using the message length and the average writing speed.

For simplicity, the typing speed is set to a constant of 287 ms delay per character. This number was experimentally obtained by averaging the measured time to write 160 characters (i.e., a SMS) for 6 different people.

Computing the transmission energy: While replaying the sequence of messages of the test conversations, the packet traces are captured in the device of the user 1 using *tcpdump* for every test, later used for computing the energy consumption. Only the traffic of the IM application is allowed using a firewall to block the rest of the traffic.

Next, the transmission energy consumption for each test is calculated from the gathered traces using the EnergyBox. EnergyBox has been evaluated against physical energy consumption measurements showing an accuracy of 98% [26]. Given the 3G network parameters specified at operator level, EnergyBox derives the 3G states of the UE employing trace-based iterative packet-driven simulation. The RRC state machine is captured by a finite state machine that simulates state transitions using the inactivity timers or exceeding the RLC buffer thresholds. The state machine is forced to go through transitions by an iterative packet-driven simulation mechanism, which uses the inter-packet interval, the size and the direction (uplink/downlink) of the data traffic trace. The energy consumption of a given packet trace for the given network parameters is calculated by associating the UE-specific power levels with the emulated intervals in each state, and integrating them over time.

The parameters of EnergyBox are set as follows: We set the 3G network settings that correspond to the operator TeliaSonera (the operator used for the tests) measured in our local (experiment) area. The inactivity timers T_1 and T_2 are set to 4.1 s and 5.6 s respectively. The RLC buffer thresholds correspond to: $B_1^u = 1000$ and $B_2^u = 294$ bytes for uplink, and $B_1^d = 515$ and $B_2^d = 515$ bytes for downlink. The time to perform the different state transitions are set to 1.7 s, 0.65 s and 0.435 s for PCH-DCH, FACH-DCH and PCH-FACH respectively. The UE power values for the different RRC states are based on earlier measurements: DCH = 612 mW, FACH = 416 mW. We set PCH = 0 W in order to quantify only the energy spent for data transmission.

The evaluation methodology and environment are employed to analyse the potential energy savings of the message bundling and the cost of the typing notification in the next sections.

7. DOES MESSAGE BUNDLING PAY OFF?

Next, we quantify the energy savings that message bundling can provide at the cost of an introduced delay due to aggregating messages.

7.1 Energy savings and bundling results

Each test conversation is tested with the following BundleTimes: 1, 3, 5 and 7 s. The results are normalised to the base energy consumption with the message bundling disabled (32.63, 26.68, 32.81 and 17.81 Joules for Dense, Sparse, Random and Short respectively). The results are based on 3 repetitions of each unique test. Additional repetitions are run when large standard deviation is observed.

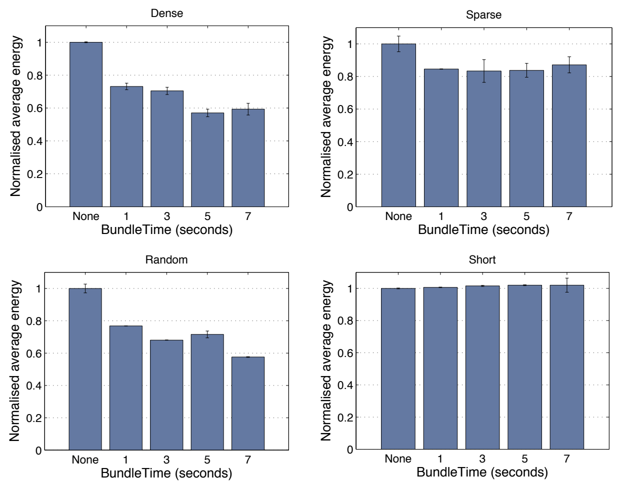

Figure 10: Normalised average energy and standard deviation for message bundling.

Energy savings: Fig. 10 shows the energy savings for the different test conversations. The message bundling provides energy savings when the algorithm successfully performs at least one bundle. The energy savings vary between the conversations, and thus we describe them separately in this section.

The energy savings due to bundling range from 27% to 43% for the Dense test conversation. Since it contains many consecutive messages and these are often close in time (i.e., the user starts writing soon after a previous message is sent), even a short BundleTime of 1 s can achieve 27% savings.

The results for Sparse are different. The message bundling achieves 16% energy savings with a BundleTime of 1 s. However, increasing the timer value does not increase the energy savings. The IMI of the consecutive messages is long for the Sparse conversation, thus the short BundleTime values do not allow performing most of the possible bundles.

Regarding the Random conversation, the energy savings range from 24% to 42% for the different BundleTimes. The shortest timer provides again significant energy savings.

There is only one consecutive message in the Short conversation. No bundle is created since its inter-message interval is longer than the BundleTimes, and the message length is short (i.e., the typing time is short).

Bundling results: Tables 2 and 3 show the achieved number of bundles and the messages per bundle for the different BundleTimes using the Dense and Sparse conversations. *Possible bundles* refers to the number of distinct bundles where each bundle is a sequence of consecutive messages with the maximum length appearing in the conversation.

Table 2: Number of bundles and bundles per message for the Dense conversation.

BundleTime (s)	Bundles	Messages per bundle
1	3	3, 3, 2
3	3	4, 3, 2
5	4	5, 3, 3, 2
7	4	5, 3, 3, 2
Possible bundles	4	5, 4, 3, 2

For Dense, the BundleTime of 1 s creates 3 bundles out of the 4 possible bundles (i.e., 4 groups of consecutive messages). Only one more message is aggregated for the 3 s BundleTime. All the possible bundles are performed when the BundleTime is increased to 5 s achieving the maximum energy savings. Similar results are obtained for 5 and 7 s, i.e., no additional messages are bundled even increasing the BundleTime.

Table 3: Number of bundles and bundles per message for the Sparse conversation.

BundleTime (s)	Bundles	Messages per bundle
1	1	3
3	2	2, 3
5	2	2, 3
7	2	2, 3
Possible bundles	3	2, 3, 3

For Sparse, the BundleTime of 1 s leads to a single bundle out of the 3 possible bundles. The other BundleTimes only create an additional bundle of 2 messages.

Protocol overhead: The message bundling also provides less TCP/IP header overhead. The number of packets sent in the Random conversation was reduced from 40 to 16 when using a BundleTime of 7 s. Thus, the messages are sent in the same packet reducing the TCP/IP overhead.

To sum up, our results show that even a short BundleTime can lead to significant energy savings. The next section studies the potential drawback of bundling.

7.2 Message delay

Even though the bundle technique is desirable from the energy perspective, one needs to also consider its negative impact on per-message delay. The bundle technique delays each message by a minimum of the BundleTime value. The delay of the held messages increases when the user continues typing a consecutive message since the BundleTimer is restarted every time the user types a character.

The per-message delay is computed for the Dense and Sparse test conversations for different BundleTimes. The application is instrumented in order to obtain the delay between the moment of pressing the sending button and the time that the message is actually sent to the network. The minimum delay for each BundleTime is the BundleTime itself, representing the case that a message was not bundled. The bundled messages are typically delayed more than the minimum, depending on the IMI, the BundleTime and the number of characters in the next message.

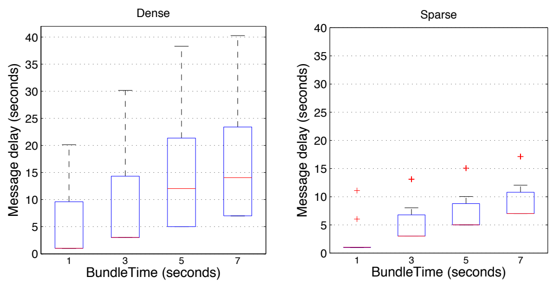

Figure 11: Box plot of the delay experienced by the different messages for Sparse and Dense conversations.

Fig. 11 shows the delay introduced by bundling in the Dense and Sparse conversations. The median delay for the Dense conversation with the BundleTimes 1 and 3 s is the BundleTime since most messages are just delayed by the minimum delay. These are the last messages added to a bundle or the messages that were not bundled. As expected, the bundled messages experience greater delay.

Fig. 11 shows that the maximum delay for the BundleTimes 1 and 3 s is 20 and 30 seconds in the Dense scenario. The maximum delay for 3 s BundleTime increases because it bundles an additional message. The more messages in the bundle, the higher is the maximum delay of the first bundled message. However, increasing the BundleTime leads to a great increase of the per-message delay, especially the messages that are bundled. In Dense, even though the BundleTimes 5 and 7 s form exactly the same bundles, the messages experience an extra delay with no extra energy saving.

The results for the Sparse conversation show that the median delay is the minimum experienced delay. The long IMI of the Sparse conversation make the bundle technique to create a single bundle of 3 messages for the 1 s BundleTime (the outliers of 6 and 11 s in Fig. 11), and an additional bundle of 2 messages for the rest. Thus, the per-message delay is shorter due to the smaller number of bundles than for the Dense conversation.

Comparing the energy savings and the introduced per-message delay, we observe that a short BundleTime of 1 s leads to significant energy savings while not causing huge delays. This indicates that simply keeping track of the user typing is enough for a simple bundle policy. Longer BundleTime values can increase the energy savings for dense conversations at the cost of higher delay. However, Sparse conversations with sporadic messages should not use long timers.

Finally, the message delay does not always have a negative impact on the QoE. When the user is not engaged in an active conversation, the reception of the non-delayed messages or a bundle of delayed messages can be argued to be the same. However, from the energy perspective, the latter drastically reduces the energy consumption.

8. COST OF TYPING NOTIFICATION

Aggressive notification policies can lead to wasting energy. This section studies the additional energy cost incurred by the typing notification functionality for the different test conversations. The results are based on 3 repetitions of each unique test. An additional 2 repetitions are run if large standard deviation is observed in the results.

For each conversation, we compute the energy consumption with the typing notification functionality disabled as a baseline. We select 3, 5 and 10 s as NotifyTime values for the different tests. These values are representative of the parameters used in real typing notification mechanisms.

Since the focus is on the additional cost, we normalise all the values to the average energy consumed by the baseline for each conversation (30.65, 26.99, 33.36 and 18.47 Joules for the Dense, Sparse, Random and Short respectively), i.e., the energy cost with no typing notification.

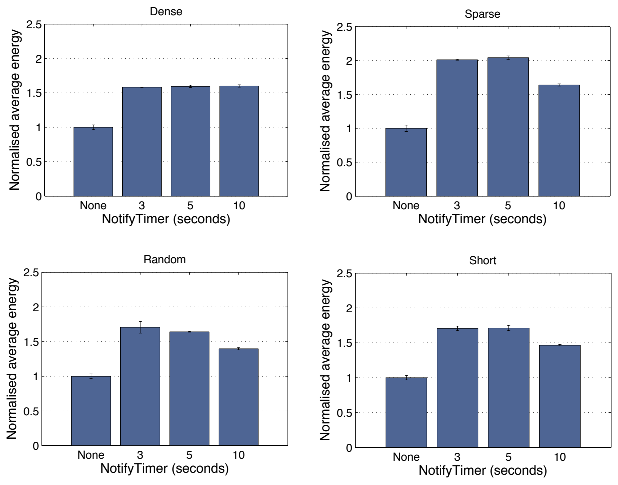

Figure 12: Normalised average energy and standard deviation of the typing notification feature for the different conversations.

Fig. 12 shows that in general the additional energy cost of the typing notification feature is large, varying from 1.4 to 2 times the base energy consumption (40-104% more energy). Since the functionality sends packets whenever the user is typing, it keeps the 3G interface in an active state for almost the whole duration of the test. The results for each conversation are explained next.

For the Dense conversation, the energy consumption for the different NotifyTime values is similar. Even though a longer NotifyTime implies sending fewer notify packets, when the device enters a high energy RRC state, sending more data does not incur higher energy cost. Most of the energy consumption is due to reseting the inactivity timers by the operator, and thus the message pattern of the conversation greatly influences the consumed energy. Since the conversation is dense, the impact of the typing notification is not as high as in the Sparse conversation.

The additional energy cost for the Sparse conversation is higher than the Dense one. The Sparse conversation has periods of time where the device is in PCH between message sendings. However, the typing notification messages greatly reduce the idle time. Since the 3 and 5 s NotifyTimes are shorter than the inactivity timer T_2, the T_2 timer is reset before it has expired. When the NotifyTime is 10 s, T_2 expires, which results in higher idle time. However, the energy cost is still 64% more than the baseline. The Sparse conversation is characterised by having slightly longer messages, and thus the time to type for the user is longer and more notify packets per message are sent.

The results for the different NotifyTimes for the Random and Short are similar to the Sparse conversation. The Short and Random conversations contain mostly short messages. When NotifyTime is 10 s, less notify messages are sent since the user spends less time typing the messages.

To sum up, the typing notification functionality results in a high additional cost for an IM application. Even though a longer NotifyTime avoids excessive usage of the 3G interface, the cost is still high.

9. CONCLUSION AND FUTURE WORK

When developing energy-efficient solutions for application data transmission, there is a trade-off between adding application features that perform network transmissions and energy conservation. There is a need to quantify the extra energy cost and the perceived functionality benefit from the user side.

The typing notification functionality employed by most IM applications quantified in our work was shown to have a tremendous energy cost. According to our results, this functionality can increase the energy consumption of an IM application by 40-104% from the basic message exchange functionality. Quantifying the cost of a functionality can allow the developers to rethink the need for the functionality or provide the option to dynamically enable/disable it (e.g., at low battery levels).

When the traffic is directly generated by user interaction, the network transmissions can easily result in energy waste. In our work we collected and studied an IM dataset from mobile users to create a better understanding of the user inputs that trigger network transmissions. Based on our study with collected real usage data, we observe that IM application users currently tend to write consecutive messages, which increases the active time of the wireless interface.

We show that bundling can be used to reduce transmissions while the user continues typing, and send them at one go. Our results show energy savings up to 43% depending on the message pattern of the conversation. Given the high percentage of the consecutive messages, this is a promising result. However, using longer timers for the bundle technique can lead to high delays for some messages. Thus, the bundle technique can be dynamically activated for sparse conversations, or when the receiving end of a chat is offline or away.

Our work can be extended by moving message bundling to the server side, which appears very interesting for group chats that usually have denser conversations. The current study can also be made more extensive by creating sets of synthetic data as well as employing further real traces, and confirming the trends observed above. Studying the user interaction changes given the different functionalities (e.g., adding bundling or removing typing notification) is left for future works.

More sophisticated bundling techniques are feasible leveraging knowledge of user activity (e.g., the screen is off, the user switched the focus to another application), message content parsing (e.g., conversational closings such as "talk to you later"), context information (e.g., built-in sensors) or presence information from other users. Distinguishing between periods of sparse and dense conversations is interesting to dynamically activate the bundle technique (e.g., moderately using the typing notification). Considering multimedia messages is also a future direction.

Acknowledgements

This work was supported by the Swedish national graduate school in computer science (CUGS). The authors wish to thank the support of Ericsson AB, and in particular B-O Hertz, Pär Emanuelsson and Claes Alströmer for providing the measurement kit which has been used in validating the EnergyBox and facilitating the measurement gathering phase. We would also like to thank the instant messaging users that shared their data, and the anonymous reviewers for their many insightful comments and suggestions.

10. REFERENCES

[1] BBC. Chat app messaging overtakes SMS texts, Informa says, accessed 20th February, 2014. http://www.bbc.co.uk/news/business-22334338.

[2] CNET. LG shares its plans for home appliance evolution, accessed 20th February, 2014. http://www.cnet.com/news/lg-shares-its-plans-for-home-appliance-evolution/.

[3] The Verge. WhatsApp now has over 400 million monthly users, accessed 20th February, 2014. http://www.theverge.com/2013/12/19/5228656/whatsapp-now-has-over-400-million-monthly-users.

[4] Tencent. About Tencent, accessed 20th February, 2014. http://www.tencent.com/en-us/at/abouttencent.shtml.

[5] P. K. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. N. Padmanabhan, and G. Varghese. Radiojockey: mining program execution to optimize cellular radio usage. In *Proceedings of the 18th Annual International Conference on Mobile Computing and Networking*, Mobicom '12, pages 101–112. ACM, 2012.

[6] N. Balasubramanian, A. Balasubramanian, and A. 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 conference*, IMC '09, pages 280–293. ACM, 2009.

[7] Y. W. Chung. An improved energy saving scheme for instant messaging services. In *Wireless Advanced, 2011*, pages 278–282. IEEE, 2011.

[8] Y. W. Chung. Investigation of energy consumption of mobile station for instant messaging services. In *Proceedings of the 2011 Tenth International*

Symposium on Autonomous Decentralized Systems, pages 343–346, IEEE. 2011.

[9] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3G/LTE wireless energy consumption. In *Proceedings of the 8th International Conference on Emerging Networking Experiments and Technologies*, CoNEXT '12, pages 181–192. ACM, 2012.

[10] R. E. Grinter, L. Palen, and M. Eldridge. Chatting with teenagers: Considering the place of chat technologies in teen life. *ACM Transactions on Computer-Human Interaction (TOCHI)*, 13(4):423–447, Dec. 2006.

[11] H. Holma and A. Toskala. *WCDMA for UMTS: HSPA Evolution and LTE*. Wiley Online Library: Books. John Wiley & Sons, 2010.

[12] J. Huang, F. Qian, Z. M. Mao, S. Sen, and O. Spatscheck. Screen-off traffic characterization and optimization in 3G/4G networks. In *Proceedings of the 2012 ACM Internet Measurement Conference*, IMC '12, pages 357–364. ACM, 2012.

[13] E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm. The character, functions, and styles of instant messaging in the workplace. In *Proceedings of the 2002 ACM Conference on Computer Supported Cooperative Work*, pages 11–20. ACM, 2002.

[14] H. A. Lagar-Cavilla, K. Joshi, A. Varshavsky, J. Bickford, and D. Parra. Traffic backfilling: subsidizing lunch for delay-tolerant applications in UMTS networks. *SIGOPS Operating Systems Review*, 45(3):77–81, ACM. Jan. 2012.

[15] J. Leskovec and E. Horvitz. Planetary-scale views on a large instant-messaging network. In *Proceedings of the 17th International Conference on World Wide Web*, WWW '08, pages 915–924. ACM, 2008.

[16] H. Liu, Y. Zhang, and Y. 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.

[17] L.-S. Meng, D.-S. Shiu, P.-C. Yeh, K.-C. Chen, and H.-Y. Lo. Low power consumption solutions for mobile instant messaging. *IEEE Transactions on Mobile Computing*, 11(6):896–904, June 2012.

[18] A. Pathak, Y. C. Hu, and M. 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*, EuroSys '12, pages 29–42, 2012.

[19] F. Qian, J. Huang, J. Erman, Z. M. Mao, S. Sen, and O. Spatscheck. How to reduce smartphone traffic volume by 30%? In *Proceedings of the 14th International Conference on Passive and Active Measurement*, IMC '13, pages 42–52. Springer-Verlag, 2013.

[20] F. Qian, Z. Wang, Y. Gao, J. Huang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Periodic transfers in mobile applications: network-wide origin, impact, and optimization. In *Proceedings of the 21st International Conference on World Wide Web*, WWW '12, pages 51–60. ACM, 2012.

[21] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Profiling resource usage for mobile applications: A cross-layer approach. In *Proceedings of the 9th International Conference on Mobile Systems, Applications, and Services*, MobiSys '11, pages 321–334. ACM, 2011.

[22] S. Schrittwieser, P. Frühwirt, P. Kieseberg, M. Leithner, M. Mulazzani, M. Huber, and E. R. Weippl. Guess who's texting you? evaluating the security of smartphone messaging applications. In *19th Annual Network and Distributed System Security Symposium (NDSS)*, ISOC. 2012.

[23] M. Z. Shafiq, L. Ji, A. X. Liu, and J. Wang. Characterizing and modeling internet traffic dynamics of cellular devices. In *Proceedings of the ACM SIGMETRICS Joint International Conference on Measurement and Modeling of Computer Systems*, pages 305–316. ACM, 2011.

[24] N. Vallina-Rodriguez and J. Crowcroft. Energy management techniques in modern mobile handsets. *Communications Surveys Tutorials, IEEE*, 15(1):179–198, 2013.

[25] E. J. Vergara and S. Nadjm-Tehrani. Energy-aware cross-layer burst buffering for wireless communication. In *Proceedings of the 3rd International Conference on Future Energy Systems: Where Energy, Computing and Communication Meet*, e-Energy '12. ACM, 2012.

[26] E. J. Vergara, S. Nadjm-Tehrani, and M. Prihodko. Energybox: Disclosing the wireless transmission energy cost for mobile devices. *Sustainable Computing: Informatics and Systems*, Elsevier. Accepted for publication. 2014.

[27] D. Wang, J. McNair, and A. George. A smart-NIC-based power-proxy solution for reduced power consumption during instant messaging. In *Green Technologies Conference, IEEE*, pages 1–10, 2010.

[28] C. Wilke, S. Richly, S. Götz, C. Piechnick, and U. Aßmann. Energy Consumption and Efficiency in Mobile Applications: A User Feedback Study. In *International Conference on Green Computing and Communications*, pages 134–141. IEEE, 2013.

[29] L. Xiang, J. Luo, and C. Rosenberg. Compressed data aggregation: Energy-efficient and high-fidelity data collection. *IEEE/ACM Transactions on Networking*, 21(6):1722–1735, 2013.

[30] Z. Xiao, L. Guo, and J. Tracey. Understanding instant messaging traffic characteristics. In *Proceedings of the 27th International Conference on Distributed Computing Systems*, pages 51–59. IEEE, 2007.