Characterizing Instant Messaging Apps on Smartphones

Li Zhang, Chao Xu, Parth H. Pathak, and Prasant Mohapatra
University of California, Davis CA 95616, USA,
{jxzhang, chaoxu, phpathak, pmohapatra}@ucdavis.edu

Abstract. Proliferation of smart devices has fueled the popularity of using mobile instant messaging (IM) apps at a rapid pace. While the IM apps on smartphones have become increasingly popular, there has only been a little research on understanding the characteristics of these apps. Because most of the IM apps use proprietary protocols, it is challenging to analyze their internal operations. In this work, we present a comprehensive characterization of mobile IM apps using experiments on LTE cellular network. We decompose the operations of an IM app into multiple independent states which allows us to systematically study them. We characterize the energy and bandwidth efficiency of each of the states and provide numerous insights. Our analysis reveals that typing notification feature of the IM apps is a major contributor to the energy consumption. We also find that the bandwidth efficiency of current IM apps are alarmingly poor compared to other applications such as email and web surfing. These, along with other findings, provided in this work can help improve the energy and network performance of IM apps.

1 Introduction

Recent years have witnessed a fast growing trend of using the new generation of mobile instant message (IM) applications such as WhatsApp, WeChat and Line on the smartphones. WhatsApp, for example, is ranked as the third all-time-popular Android apps in Google's Android app store [1] with a total of 590 million users in 193 different countries [5]. According to [2], the mobile IM apps have overtaken the Short Message Service (SMS) operated by cellular network carriers, with 19 billion messages sent per day compared with 17.6 billion SMS messages.

While the adoption of mobile IM apps are rapidly increasing, very little research has been done in characterizing them. This is because there are numerous challenges in characterizing the IM apps. First, compared to other types of mobile apps studied in [6,10,15,17], the IM apps involves much more user interaction such as typing, reading and user notifications. This makes the automated characterization extremely difficult. The new set of features (e.g. typing and read notifications) offered by the IM apps are much more complex compared to the traditional SMS services. Also, there is a lack of transparency in the application layer protocols used by the popular IM apps. Most of the current IM apps either implement their own protocol or modify existing standard such as XMPP to customize them. This makes it even more difficult to understand the underlying operations of the apps.

In this work, we present a comprehensive characterization of the popular IM apps for smartphones using experiments on LTE cellular network. We address the challenges listed above by dissecting the operations of IM apps into many different states and then evaluate the energy and the network efficiency of each of them. Some of the main insights provided by our study are as follows:

- We find that sending and receiving typing notifications are major contributors to the total energy consumption when the IM app is running in the foreground. Many IM apps use frequent periodic typing notification messages which result in very poor energy efficiency.
- Today's IM apps have extremely low bandwidth efficiency (average amount of traffic per one character of user message). This is true even when the app is running in the foreground and has minimal requirement of maintaining the "online presence". This shows that while XMPP-like IM protocols offer efficient ways of maintaining "online presence", the current IM apps show poor network efficiency when running in the foreground.
- Because users spend significant amount of time on IM apps compared to other types of apps, simply switching to darker graphical interface can yield surprising energy benefits.
- When the IM apps are running in background, the method used to notify the user about incoming message has a significant impact on the energy consumption.

The rest of paper is organized as follows. We describe the experimental setup and the data collection in Section 2. The foreground and the background characterization results are presented in Section 3 and Section 4 respectively. Then we discuss the related works in Section 5. Section 6 concludes the paper.

2 Data Collection and Methodology

In this section, we first provide the details of data collection for different IM apps. We represent the operations of an IM app using a state transition diagram. For each of the states, we will test 5 most popular IM apps, and profile the energy consumption and the network traffic generated.

2.1 State Transitions in IM App Usage

As shown in Fig. 1, the operations of an IM app can be divided into 6 distinct states. When the users are in a conversation, the IM app runs in the foreground, occupying the entire screen. When the users are using another app or when the screen is turned off, the IM app runs in the background but still keeps maintaining connections with its remote servers.

Foreground: When the IM app is in the foreground, the user is considered to be "in conversation". There are two "in conversation" states.

- In Conversation Sending (ICS): The ICS state is defined as the period from when the user starts her typing of the message to the time when the read notification is received. In this state, there are 4 functions: type, send typing notification, send message and receive read notification.

- In Conversation Reading (ICR): The ICR state is from receiving the typing notification to sending the read notification. This state has 3 functions: receive typing notification, receive and display message, send read notification.

Background: There are four background states.

- Background Idle with Screen On (BION): BION is a state that the IM app is running in the background while neither occupying the screen nor getting any incoming message. This state has only 1 function: keep maintaining the on-line presence with the server.

- Background Receive with Screen On (BRON): The BRON state is from the time when message starts to arrive to the time when its notice is displayed to the user. This state has 2 functions: receive message and display notice. This state ends before the user takes any action for the received message, therefore the IM app will not send out a read notification.

- Background Idle with Screen Off (BIOFF): The BIOFF state is the period when the IM app is idly listening in the background and the screen is off. Similar to BION state, the BIOFF state also has only 1 function: keep maintaining the on-line presence.

- Background Receive with Screen Off (BROFF): This state starts when the message arrives and the screen is off. This state ends once the user is notified by some form of notification either using sound, vibration or screen turn-on.

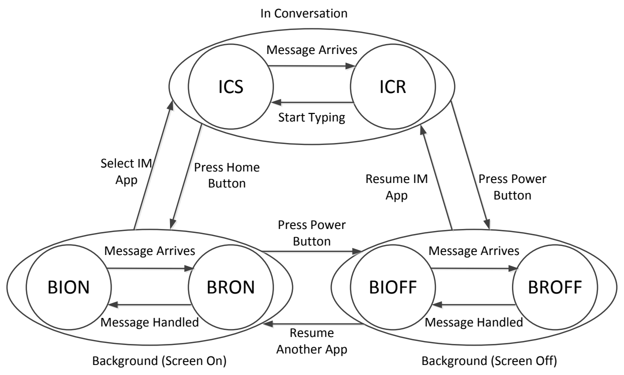


Fig. 1: The state transition diagram of an IM app usage

2.2 Experiment Settings

We select the top 5 mobile IM apps in terms of the number of users by the end of 2013. The names and the statistics of the selected apps are listed in Table. 1. Our experimental setup consists of a Samsung Nexus S smartphone (running Android 4.1.2), a Macbook Air, a Dell Latitude E5430 laptop and a Monsoon Power Monitor. We install *tcpdump* on the phone to capture the network traffic. The Macbook Air is used as the phone's SSH console. The Monsoon Power Monitor is employed to measure the power consumption of the smartphone, and the Dell laptop serves as the power monitor console. The sampling frequency of the power monitor is set to 5 KHz.

We conducted all the experiments on AT&T's cellular network data connection. We also turn off the WiFi and Bluetooth radios and fix the volume, brightness, vibration and keystroke feedback settings to avoid any unintended impact on measured energy. To turn off any additional background services on

Apps	**Mobile Users**	**Covered Countries**	**Originated From**	**Icon**
WeChat	600 million	$\sim$ 200	China	
WhatsApp	590 million	$\sim$ 100	USA	
Facebook Messenger	300 million	Unknown	USA	
Line	300 million	193	Japan	
Viber	200 million	193	Israel	

Table 1: List of selected IM apps; Number of users data from [5]

Android, we limit the number of background processes to one and use "Advanced Task Killer Pro" app to kill any additional running processes.

2.3 Methodology and Metrics

To get a comprehensive view of the characteristics of the selected IM apps, we test all the 6 states of the apps, by using a set of the most commonly used IM messaging literacy among college students [7]. In [7], the authors listed the taxonomy of the IM conversation topics. For example, the 5 most popular conversation topics are: emotional support, fictional people, video games, computers and shared interests. We picked one conversation in each kind of the popular topics from the typical examples concluded in [7] and created a database of 70 messages. The length of the messages varies from 4 characters to as many as 125 characters, where the characters may include letters, punctuation marks and metadiscursive markers. To reduce the effect of randomness, the typing of each message in each run of the experiments is repeated 20 times to calculate an average value. We repeat the experiments for two different users to eliminate any user-specific typing characteristics.

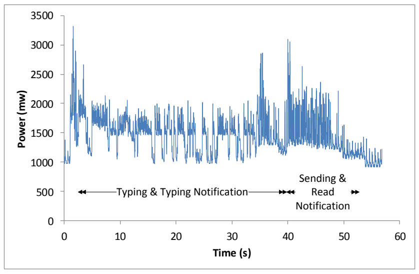
The performance of the IM apps in the state with sending/receiving activities are mainly evaluated by two metrics: (i) Energy efficiency: energy consumption per character sent/received (Joule/character) and (ii) Bandwidth efficiency: the amount of network traffic generated per character sent/received (byte/character). In the idle listening states, since there are no user intended messages, we will use the average energy consumption per hour (J) and the average network traffic per hour (KByte) as the evaluation metrics.

3 In Conversation Sending/Receiving (ICS/ICR)

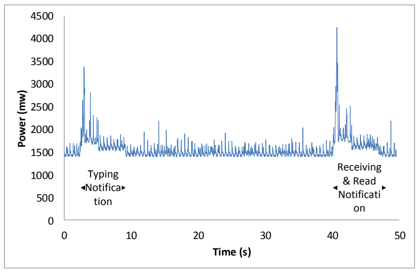
We conducted a total of 12,600 runs of experiments by manually typing, and collected 2.4 GB of energy and network traffic traces. From the network traces, we observed that all the 5 selected IM apps are built on the client/server architecture, where the message sender and the message receiver communicate indirectly through a certain number of servers. Although following the same architecture, the application layer protocols used by each app are quite different. By linking the server port number with the registry of Internet Assigned Numbers Authority (IANA) [8], we found that WeChat, WhatsApp, FB Messenger, Line and Viber use commplex-main, XMPP, HTTPS, SSL and Virtual Reality Modeling Language (VRML) [14] respectively.

As shown in Fig. 2, the ICS state can be divided into two phases: 1) typing the message & sending typing notification, and 2) sending the message &

receiving the read notification.Correspondingly, the ICR state is also consisted of two phases: 1) receiving the typing notification, and 2) receiving the message & sending the read notification. Since the typing of a long message needs considerable amount of time, we can observe a time gap between the first and the second phase of the ICR state. During the time gap, the radio will be tuned to the paging channel (PCH) state to save energy.



(a) Sending a Message with 120 Characters



(b) Receiving a Message with 120 Characters

Fig. 2: Examples of Energy Traces of In Conversation States (WhatsApp)

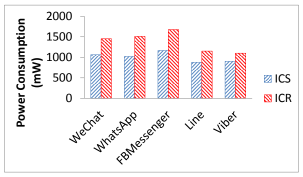
3.1 Energy Characterization

The energy consumption of "in conversation" states can be attributed to two factors: (1) the Graphical User Interface (GUI) and (2) user operations such as typing or sending messages etc.

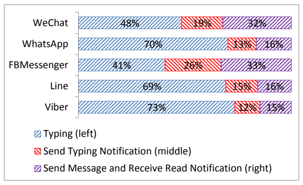
GUI The average values of the energy consumption of the GUIs of the IM apps are shown in Fig. 3(a). It is observed that the GUIs of the ICR states always consume more energy than the GUIs of the ICS states (36.3% more on an average). This is because the conversation windows of the IM apps are usually in brighter colors, while the default keyboard background of Android is in darker color. In the ICR state, the conversation window usually occupies the entire screen; while in the ICS state, the dark keyboard will occupy about half of the screen which reduces the overall energy consumption. Therefore the GUIs of ICR state will consume more energy than the GUIs of ICS state.

Since the energy consumption of the display is highly dependent on the hardware, there is only a little that can be done from the app development perspective. We observe that Line and Viber (refer Fig. 3(a)) consumes much less energy in ICR and ICS states simply due to the fact that their GUIs use darker colors. Because users spend a large amount of time on the IM apps (very high user residence time [17]), it is advisable to incorporate such modifications. We observe that the GUI consumption of each app in each state is more or less constant (coefficients of variance laying in the range of $(0.0053, 0.0228)$), hence we deduct the GUI energy from the energy measurements shown in the rest of this paper.

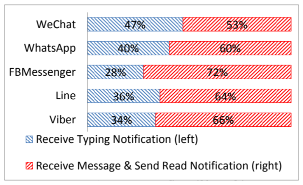
ICS and ICR User Operations The characteristics of the energy consumption related to user operations are shown in Fig. 3(b) and Fig. 3(c). In the ICS



(a) The Energy Consumed by The GUIs



(b) ICS Energy Consumed by User Operations



(c) ICR Energy Consumed by User Operations

Fig. 3: The Factors of Energy Consumption

state, we can see the typing message and sending typing notification time phase consumes much (308% on average) more energy than the energy consumed by sending the message and receiving the read notification. However, on the receiving side, the difference of the energy consumed by the user operations in the two time phases is relatively small (40% on average).

Energy for Typing: We first turn off the radio and simply measure the energy of typing. We observe that over 60.2% of the energy cost is attributed to typing in the ICS state.

Energy for Typing Notifications: We observe that sending the typing notification in ICS state consumes as much energy as sending the actual message and receiving the read notifications combined. In the ICR state, receiving typing notification consumes as much as 37% (average for all 5 apps) of state's total energy consumption. The high energy consumption is due to the fact that however small the typing notification message is, it requires the radio interface to be turned on. This shows that sending and receiving typing notifications is a major factor of energy consumption (often comparable to sending and/or receiving the actual message). This means significant amount of energy can be saved by simply turning off the typing notifications. This also calls for a more energy efficient solution for enabling typing notifications.

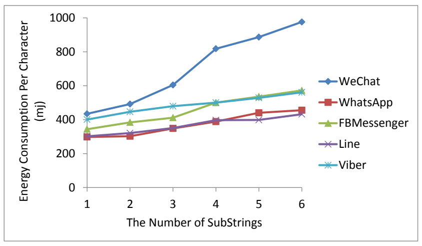
Energy for Read Notifications: Because the sending and receiving read notification is submerged in receiving and sending message respectively, it is difficult to isolate the energy consumption of the read notification. However, we expect the energy consumption of the read notification to be much lesser than that of typing notification. This is because the read notification is only sent once while the typing notification involves many messages (continuously based on when user starts and stops typing until the message send button is pressed). Also, because sending/receiving read notification is mostly submerged with receiving/sending the message, no separate radio wake up is necessary, further reducing its energy overhead.

Energy Efficiency We now present the results about per character energy consumption as defined in Sec. 2.3. To understand this, we compare the energy consumption for many short messages to fewer long messages. The size of the complete message is chosen to 120 characters which is divided into substrings, each of which is sent individually. As shown in Fig. 4, we consider 1 to 6 as possible number of substrings. When number of substrings is 1, it means that

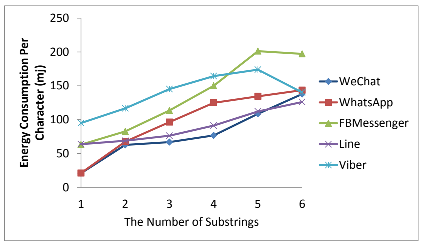
Apps	Number of TCP Connections	Number of Packets	Average Packet Size (byte)	Duration (second)
WeChat	1	38	61.4	33.54
WhatsApp	1	14	110.9	21.35
FB Messenger	1	28	166.4	22.42
Line	1	12	87	20.74
Viber	1	26	108.5	22.39

Table 2: ICS: The Average Traffic Statistics of 30 Character Messages

the entire 120 character message is sent at once. On the other hand, when the number of substrings is 6, a total of 6 messages are sent separately each of which is of 20 characters. Fig. 4 shows energy consumption per character when different number of substrings are sent.



(a) ICS Energy Consumption Per Character



(b) ICR Energy Consumption Per Character

Fig. 4: Energy Consumption Per Character

As shown in Fig. 4(a), the energy spent on sending each character increases as the length of the substrings decreases. This is mainly caused by the overhead of sending typing notifications because an IM app needs to tune its radio to dedicated channel (DCH) state and also suffers the tail energy overhead in the Forward Access Channel (FACH) state. We can also observe the energy efficiency of WeChat is much lower compared to other apps. This is because WeChat aggressively sends typing notifications every 2 seconds. Since typing many shorter messages takes much more time combined than typing one long message, number of typing notifications increase sharply for WeChat, resulting in sharp increase in energy consumption. To further validate this, we present network traffic statistics in Table. 2. It shows that WeChat sends more smaller packets for typing notifications. On the other hand, WhatsApp and Line send fewer packets due to fewer typing notifications which also reflects in their per character energy efficiency in Fig. 4(a).

In the ICR state, the energy consumption also follows the same trend that many shorter messages consume more energy as shown in Fig. 4(b). However, we observe some anomaly in case of FB Messenger and Viber where many shorter messages (no. of substrings = 6) are more energy efficient compared to fewer medium sized messages (no. of substrings = 5). This is because both these apps *delay* sending the typing notifications. This allows the typing notification to be received almost at the same time (compared to Fig. 2) with the actual message,

Apps	**Number of TCP Connections**	**Number of Packets**	**Average Packet Size (byte)**	**Duration (second)**
WeChat	1	20	280	10.3
WhatsApp	1	12	78.7	11.5
FB Messenger	2	26	235.3	16.4
Line	1	8	187.5	12.4
Viber	1	14	122.4	22.6

Table 3: ICR: The Average Traffic Statistics of 30 Character Messages

which eliminates additional radio wake up and saves energy. This shows that if the typing notification can be delayed towards sending the actual message, it is possible to reduce the energy overhead of sending/receiving the typing notifications, especially for small length messages.

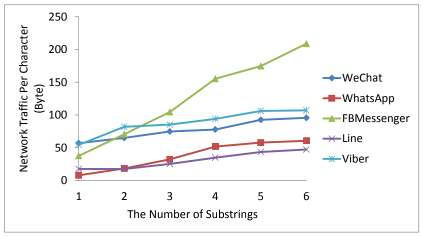
Findings: (1) Sending and receiving typing notification is a major contributor to the total energy consumption of ICR and ICS states. IM apps which periodically send the typing notifications suffer from very high energy consumption. Because most of the IM messages are small in length, delaying the sending of typing notification can save significant energy. Also, an adaptive scheme should be designed that can control when to send the typing notification depending on the length of user's input message. Such a scheme can achieve the correct balance between usefulness of typing notifications and their energy consumption. (2) Because user's residence time on IM apps are much longer compared to other types of apps, simply switching to darker GUI can yield surprising energy benefit.

3.2 Bandwidth Efficiency

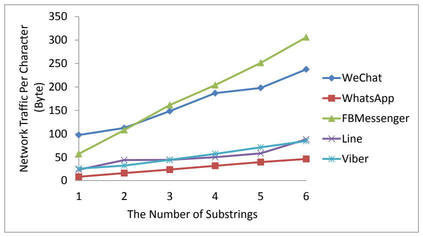
We now analyze the bandwidth efficiency (amount of traffic generated per character sent/received) of the IM apps and present the results in Fig. 5(a) and Fig. 5(b). This helps us to understand how much traffic the IM apps generate compared to the amount of useful information (instant message) exchanged. It is observed that network traffic per character is different when receiving or sending the same message. This is expected given that all the apps use client-server architecture and the sent message is first processed at the server before it is delivered to the receiver. It is also observed that network traffic per character is much higher on the receiving side compared to the sending side.

We observe that FB Messenger has the worst bandwidth efficiency for both sending and receiving sides in most cases. On the other hand, WhatsApp and Line achieve very high bandwidth efficiency compared to other apps. Due to the unavailability of their internal design, application layer protocol customization etc., it remains inconclusive why certain apps achieve high or low bandwidth efficiency.

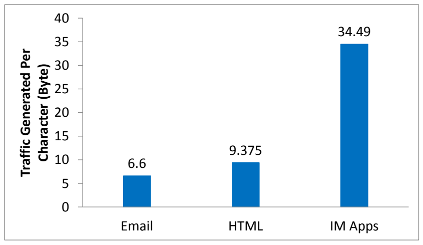
Comparison with Other Types of Applications We now compare the bandwidth efficiency of IM apps to other kind of applications. We first construct a set of emails and plain HTML pages with the same set of messages tested on the IM apps. For email, we measure the amount of traffic generated by Google Mail and the size of the actual emails. For HTML, we set up a web-server which holds a plain HTML page (without any images) and connect it via a client to measure the traffic and the size of HTML page. The bandwidth efficiency of Email and



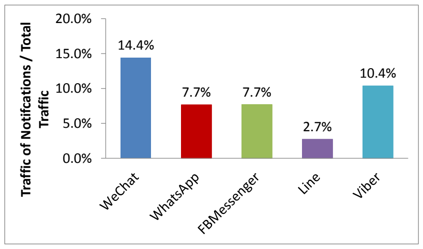
(a) ICS Network Traffic Per Character



(b) ICR Network Traffic Per Character



(c) The Bandwidth Efficiency of Different Applications



(d) Ratio of Traffic (Byte) of Notifications to The Total Traffic

Fig. 5: The Bandwidth Consumption Statistics of In Conversation States

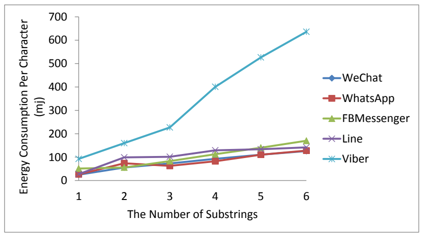
HTML are compared with IM apps in Fig. 5(c). As we can observe, IM apps have extremely poor bandwidth efficiency which shows that even the modern protocols such as XMPP (used by WhatsApp) are not bandwidth efficient.

Traffic due to Typing and Read Notifications It was observed in Section 3.1 that typing notifications are a major contributor to energy consumption. We now evaluate how much network traffic is generated due to the typing and read notifications. Fig. 5(d) shows the ratio of traffic due to notifications to the total traffic. We observe that the ratio is small for most of the applications. This way, the actual traffic due to notifications is low, however, because the notifications are sent/received using many small packets (Table. 3), it causes frequent radio wake up and results in poor energy efficiency.

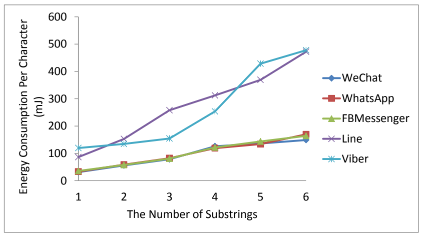
Findings: (1) The IM apps have extremely poor bandwidth efficiency compared to other applications such as email and web-surfing. Modern IM protocols such as XMPP which are optimized to reduce traffic in background states demonstrate the same poor level of bandwidth efficiency in the foreground states. Further improvements are necessary to improve the network performance of instant messaging apps and protocols. (2) Typing notification which is a major contributor in energy consumption does not introduce proportionally high network traffic.

4 The Background States

The performance of the IM apps running in the background is now characterized. We show the corresponding results in Fig. 6 - 8. Similar to the ICR state, we can also observe the energy efficiency of the background receiving decreases if the the length of the messages decreases, as shown in Fig. 6. However, the reasons behind the phenomenons are quite different. In the background receiving states, we did not observe any typing notification nor read notification from the network traces. The main cause of the energy efficiency reduction is the overhead of notifying the users through certain methods, e.g. banner size notification, pop-up window and icon label. In the BRON state, Viber uses pop-up window, while in the BROFF state, both Line and Viber use pop-up window. The pop-up window results in significant increase of energy consumption for these two apps as seen in Fig. 6.



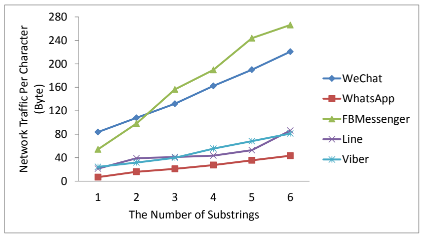
(a) BRON Energy Consumption Per Character



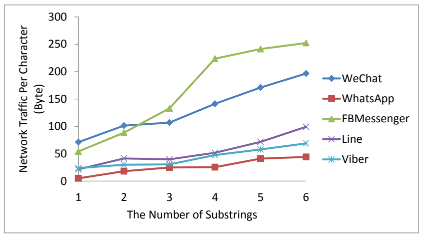
(b) BROFF Energy Consumption Per Character

Fig. 6: The Energy Efficiency of Background Receiving

Comparing Fig. 7 and Fig 4(b),we can observe that the bandwidth consumption of the background receiving follows the same trend but is slightly lower than the bandwidth consumption of the ICR state, since there are no typing notifications and read notifications. On average, the BRON and the BROFF states consume 9.6% and 8.5% more bandwidth than the ICR state.



(a) BRON Bandwidth Consumption Per Character

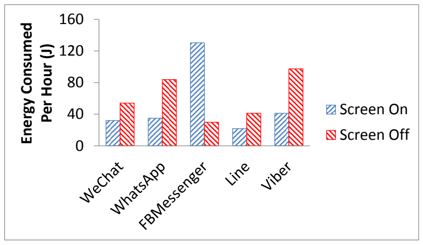


(b) BROFF Bandwidth Consumption Per Character

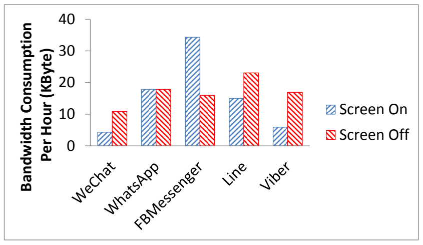
Fig. 7: The Bandwidth Consumption of Background Receiving

The statistics of the idle listening states are shown in Fig. 8. The main function of the idle listening states is to communicate periodically with the server to maintain the online presence. Therefore the number and the frequency of exchanging "keep online" packets are the main factor affecting the energy consumption. From the results, we can observe the implementation of the "keep online" function is comparatively more energy efficient. For example, considering the 1,500 mAh battery of Nexus S, the FB Messenger in BION state can have 172 hours of standby time.

Findings: (1) Energy efficiency of different IM apps in background receiving states depends mostly on how the app notifies the user about incoming message.



(a) Energy Consumption of Idle States



(b) Bandwidth Consumption of Idle States

Fig. 8: The Statistics of Idle States

Apps that use pop-up window notification consume drastically more energy than the apps using banner notification or icon label. This sheds light on the potential to improve energy efficiency by simplifying the user notification process. (2) With improved methods of maintaining "online presence" in today's IM protocols such as XMPP, the energy and bandwidth efficiency of idle states are comparatively better than other states.

5 Related Work

Traditional Messaging Services: There is a limited amount of prior work on characterizing the performance of IM apps on smartphones. PC-based IM apps (AIM and MSN) were characterized in [16] where authors studied network traffic related characteristics. Similarly, [9] characterized the users' conversation styles of IM apps in workplace, by analyzing the SMS messages exchanged through AT&T's cellular network. Note that different from both these efforts, we have attempted to characterize smartphone IM apps which have revolutionized the way people connect in today's era.

App Profiling: There has been multiple research works on developing methods to profile smartphone apps in general. This includes multi-layer profiling tool *ProfileDroid* [15], *Application Resource Optimizer (ARO)* [12], energy measurement tool *eprof* presented in [11] and third-party API resource usage measurement tool *API Extractor (APIX)* presented in [18]. Different from these generic profiling tools, our focus in this work is to understand the network and energy characteristics specific to the IM apps.

Mobile IM Apps: Considering the research specific to mobile IM apps, [3] and [4] modeled user's residence time on IM apps and typical message arrival rate. Based on these models, they derived energy consumption models of IM apps. The provided model, however, only provides a high-level coarse-grained behavioral analysis which is independent of the operation of the underlying IM app. In this paper, our focus is on the operations of different IM apps. In other related work [13], the authors showed the energy consumption of IM apps can be reduced by message bundling. To evaluate their bundling algorithms, the authors implemented a customized IM app and developed a software tool *Energy Box* to estimate the energy consumption of sending/receiving instant messages by analyzing the tcpdump traces. Note that such techniques to improve energy efficiency of IM apps are in line with our effort to quantify the energy consumption of popular IM apps.

6 Conclusions

By decomposing the operations of IM apps into 6 states, we characterized the energy and the bandwidth efficiency of IM apps. We also analyzed various operations of the IM apps, e.g. typing notification, read notification, sending/receiving messages. Our analysis revealed there is still plenty of improvements necessary in the IM apps especially in the "in conversation" and the "background receiving" states to improve their energy and bandwidth efficiency. However, we observe that the background idle states already have comparatively high energy and bandwidth efficiency.

References

1. AppBrain, http://www.appbrain.com/stats/.
2. BBC, http://www.bbc.com/news/business-22334338.
3. Y. W. Chung, "Investigation of energy consumption of mobile station for instant messaging services," ser. ISADS'2011, pp. 343–346.
4. ——, "An improved energy saving scheme for instant messaging services," ser. WiAd'2011, pp. 278–282.
5. C. Clifford, "Top 10 apps for instant messaging," *Entrepreneur*, Dec 11, 2013.
6. H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," ser. IMC '2010, pp. 281–287.
7. C. Haas and P. Takayoshi, "Young people's everyday literacies: The language features of instant messaging," *Research in the Teaching of English*, vol. 45, no. 4, pp. 378–404, May, 2011.
8. I. A. N. A. (IANA), https://www.iana.org/assignments/.
9. E. Isaacs, A. Walendowski, S. Whittaker, D. J. Schiano, and C. Kamm, "The character, functions, and styles of instant messaging in the workplace," ser. CSCW '2002, pp. 11–20.
10. S.-W. Lee, J.-S. Park, H.-S. Lee, and M.-S. Kim, "A study on smart-phone traffic analysis," ser. APNOMS'2011, pp. 1 –7.
11. A. Pathak, Y. C. Hu, and M. Zhang, "Where is the energy spent inside my app?: fine grained energy accounting on smartphones with eprof," ser. EuroSys '2012, pp. 29–42.
12. F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck, "Profiling resource usage for mobile applications: a cross-layer approach," ser. MobiSys '11. ACM, 2011, pp. 321–334.
13. E. J. Vergara, S. Andersson, and S. Nadjm-Tehrani, "When mice consume like elephants: Instant messaging applications," ser. e-Energy '2014, pp. 97–107.
14. VRMLSite, http://www.vrmlsite.com.
15. X. Wei, L. Gomez, I. Neamtiu, and M. Faloutsos, "Profiledroid: multi-layer profiling of android applications," ser. Mobicom'2012.
16. Z. Xiao, L. Guo, and J. Tracey, "Understanding instant messaging traffic characteristics," ser. ICDCS'2007, pp. 51–51.
17. Q. Xu, J. Erman, A. Gerber, Z. Mao, J. Pang, and S. Venkataraman, "Identifying diverse usage behaviors of smartphone apps," ser. IMC'2011, pp. 329–344.
18. L. Zhang, C. Stover, A. Lins, C. Buckley, and P. Mohapatra, "Characterizing mobile open apis in smartphone apps," in *IFIP Networking Conference' 2014*, pp. 1–9.