SPS: an SMS-based Push Service for Energy Saving in Smartphone's Idle State

Erich Dondyk
Amazon.com, Inc.
Cambridge, MA

Omar Nakhila
University of Central Florida
Orlando, Florida 32816

Cliff C. Zou
University of Central Florida
Orlando, Florida 32816

***Abstract*—Despite of all the advances in smartphone technology in recent years, smartphones still remain limited by their battery life. Unlike other power hungry components in a smartphone, the cellular data and Wi-Fi interfaces often continue to be used even when the phone is in its idle state in order to accommodate background (necessary or unnecessary) data traffic produced by some applications. In addition, bad reception has been proven to greatly increase energy consumed by the radio, which happens frequently when smartphone users are inside buildings. In this paper, we present a Short message service Push based Service (SPS) system to save unnecessary power consumption when smartphones are in idle state, especially in bad reception areas. First, SPS disables a smartphone's data interfaces whenever the phone is in idle state. Second, to preserve the real-time notification functionality required by some apps, such as new email arrivals and social media updates, when a notification is needed, a push server will deliver a wakeup text message to the phone (which does not rely on data interfaces), and then SPS enables the phone's data interfaces to connect to the corresponding server to retrieve notification data via the normal data network. Once the notification data has been retrieved, SPS will disable the data interfaces again if the phone is still in idle state. We have developed a complete SPS prototype for Android smartphones. Our experiments show that SPS consumes less energy than the current approaches. In areas with bad reception, the SPS prototype can double the battery life of a smartphone.**

I. Introduction

Smartphones have penetrated the technology market at a staggering rate. In fact, it is estimated that currently more people own smartphones than personal computers throughout the world [1]. However, despite of the tremendous functionality smartphones provide, they still remain plagued by a short battery life.

One of the main sources of energy consumption in smartphones is the wireless radio. Specifically, the data network interfaces used for cellular data, such as 3G and 4G, and Wi-Fi. The data interfaces inherently need a significant amount of energy when transmitting and receiving data. In addition, 'tail power phenomenon' introduces additional energy consumption for every retransmission [2]. The cause and possible solutions to these internal sources of energy consumption have been explored in other works [3] [4] [5]. However, there are also two external factors that can significantly contribute to the total amount of energy the data interfaces consume: bad reception and data traffic in idle state.

Bad reception introduces a series of side effects that greatly increase the amount of energy consumed by data interfaces.

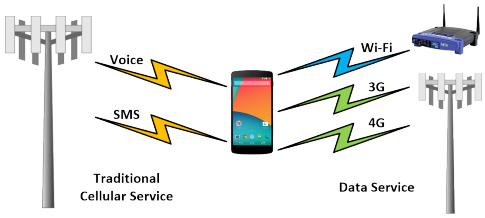

Fig. 1. The communication interfaces of a smartphone. The data interfaces, such as 3G, 4G, and Wi-Fi, are among the components of the smartphone that consume the most energy. Our proposed energy-saving scheme exploits the traditional cellular short message service (SMS) to facilitate real-time data notification service that currently solely relies on the data service channels.

First, smartphones experiencing bad reception frequently dissociate and reassociate with base stations or switch between different cellular technologies such as 3G and 4G. This frequent switching back and forth, known as the ping-pong effect, produces additional energy costs. Secondly, bad reception often causes retransmissions at the physical and transport layers which also produces additional energy costs. Finally, the smartphone radio is design to adjust their parameters when experiencing bad reception in order to increase throughput and decrease bit error. These adaptations generally result in lower data rates and higher transmission powers, both of which consume more energy. With the average smartphone experiencing bad reception 47% of the time [6], the aggregate impact of these side effects can significantly reduce the battery life.

An average smartphone user interacts with his smartphone for only 58 minutes per day [7]. The remaining of the day smartphones are usually in the idle state. While in the idle state, components such as the display and the processor are normally turned off or placed in low power states to conserve energy. However, the data interfaces often continue to be used during the idle state to accommodate the data traffic generated by some applications. In fact, study [6] shows that 19% of all the traffic generated by a smartphone is generated during the idle state. Most of this idle state traffic is unnecessary because it provides no immediate functionality to the user and it is often the result of careless application design.

On the other hand, some of the idle state data traffic is necessary. For example, new email arrivals, social status updates, and instant messages all require real-time notification to their users. Currently, push services are the mechanism by which notifications are delivered to smartphones, which rely on long-lived TCP connections between smartphones and

the push server [8]. Such a connection serves as a "virtual circuits" by which notification data that the smartphone has not requested can be "pushed" to the smartphone. Therefore, the data interfaces in current smartphones are required to remain active even during the idle state in order for smartphones to maintain data channel to receive notifications. Such an always-on approach for data interfaces consume significant battery power during the long idle state time period.

In this paper, we remove the always-on requirement for data interfaces, and present a Short message service Push based Service (SPS) to save unnecessary power consumption when smartphones are in idle state, especially in bad reception areas. SPS disables a smartphone's data interfaces whenever the phone is in idle state—this will remove all the energy wasted in maintaining data interfaces in idle state, and remove energy consumed by the unnecessary idle state data traffic. To preserve the real-time notification functionality required by some apps, such as new email arrivals and social media updates, when a notification is needed, a wakeup short message service (SMS) message will be received by the phone, and then SPS enables the phone's data interfaces to connect to the corresponding server to retrieve notification data via the normal data network (either broadband cellular data connection or WiFi connection). Once the notification data has been retrieved, SPS will disable the data interfaces again to conserve energy during the phone's idle state.

Besides energy conservation, the proposed SPS can provide additional benefits. First, it enables a smartphone user to better control what idle state data traffic is allowed during the phone's idle state. This is especially useful for mobile users who have unlimited or cheap text messaging but limited cellular data plan. Second, SPS also enables a mobile user to control how frequent she wants to receive notifications from the push server by specifying her preference in SPS configuration. This will prevent the mobile user from being too much distracted by a burst of notifications.

Our contributions in this paper are:

• We have proposed an SMS push based scheme to conserve energy consumption when smartphones are in idle state, especially at bad reception areas. The scheme exploits the traditional cellular text messaging as a side channel in facilitating network data communication when the data interfaces are disabled during phone's idle state.

• We have developed a prototype server and an Android client that controls the data interfaces and utilizes the SMS based push service proposed in this paper. Our prototype allows us to simulate the notification traffic between a server and multiple smartphone applications. This enables us to measure the energy saving obtained using the proposed scheme. In addition, we have developed a server and client that use the push service commonly employed by current Android applications, Google Cloud Messaging (GCM) [9]. In this way, we are able to measure how our proposed SPS compares against traditional push services in term of notification delay and energy consumption.

The remaining of the paper is organized as follows. We discuss related work in Section II. Section III presents the proposed energy saving scheme. Section IV presents the evaluation of the proposed energy saving scheme, and finally Section V concludes this paper.

II. Related Work

Previous works on the impact of the cellular data interface in the battery life of smartphones have focused on the RCC power states [3] [4] [5]. These works conclude that the RCC state machine introduces significant energy inefficiencies because of the power state promotion overhead and the tail effect. These works propose modifying the inactivity timers used to transition from high power states to low power states. Our work is different because it addresses external factors, such as unnecessary idle state traffic and bad reception, that significantly contribute to the overall energy consumed by the radio.

Zhang et al. proposes E-MiLi (Energy-Minimizing idle Listening) the reduce energy consumption of Wi-Fi components [10]. By analyzing real-world Wi-Fi traffic traces, the author determines that idle listening is responsible for 60% to 80% of the Wi-Fi's energy consumption. To reduce the energy consumption during idle listening, E-MiLi reduces the clock-rate of the radio during idle listening and reverts to full clock-rate when the radio transmits or receives packets. Our work is different from Zhang's because our proposed SPS reduces the Wi-Fi's energy consumption in smartphones caused by external factors such as bad reception and unnecessary idle state traffic.

Ding et al. investigate how bad reception increases the amount of energy consumed by the smartphone radio [6]. The author developed a more accurate model of smartphone battery life. Furthermore, Ding proofs that bad reception significantly increases the amount of energy that the smartphone radio consumes. Ding concluded that buffering idle state traffic while the smartphone is experiencing bad reception could reduce energy consumption by up to 21.5%. Our approach is different from Ding's because it allows the smartphone to continue to receive notification even if there is bad reception. In addition, our approach also addresses energy consumption due to unnecessary idle state traffic.

There are several works that study energy consumption of the push mechanisms employed in smartphones. Haverinene et al. and Gupta et al. investigates how the keep-alive traffic used by push mechanisms affects the battery lifetime of smartphones in WCDMA and LTE networks respectively [11] [12]. Both authors propose modifying the RCC parameters to reduce the impact on keep-alive traffic in the battery life. Meng et al. proposes using buffers and a number of middle agents to reduce energy consumption of push mechanism used by instant message applications in smartphones [13]. Dinh et al. compares the energy consumed by an application that uses push services against an application that uses polling to retrieve unsolicited data from a server [14].

Our work is different from all the above works because, rather than fine tuning the current push mechanism employed in smartphones, we propose disabling data interfaces during phone's idle state to conserve energy and using a completely different push mechanism based on SMS channel instead of traditional data channel. This new mechanism is transparent to smartphone users, does not affect real-time notification

requirement for normal apps, and significantly saves energy consumption especially in areas with bad reception.

III. Proposed Energy Saving Scheme

In this section we describe our proposed SPS energy saving scheme. Basically speaking, it contains two parts: radio frugality policy, and SMS-based push service.

A. Radio Frugality Policy

We decrease battery impact of the wireless radio by reducing the amount of active time of data interfaces throughout the day. We achieve this by disabling the data interfaces whenever the smartphone is not being used, i.e., when the smartphone is in the idle state. And, enabling the data interfaces when the smartphone returns back to the active state. Under this radio frugality policy, the user of a smartphone continues to receive phone calls and text messages when the phone is in idle state and is able to use the data service whenever he/she uses the phone. Thus, from the user's perspective, data service appears uninterrupted. Because statistical study shows that smartphones spend the majority of the day in the idle state [7], this radio frugality policy significantly reduces the amount of time the radio is being used unnecessarily. The proposed radio frugality policy mimics the policies mobile platforms apply to other power hungry components such as the display and processor. These policies treat power hungry components as expensive resources, and thus, try to minimize the amount of time they are used.

The proposed policy saves energy by addressing two of the main sources of energy consumption in smartphone radio when the phone is in idle state. First, it reduces the impact of bad reception in the radio. Bad reception prompts the radio to consume significantly more energy regardless of whether the smartphone is in the idle or active state. This is due to the frequent dissociation and reassociation with base stations[15], and radio link power adaptation during back reception stage [6]. Since on average a smartphone experiences bad reception 47% of the day [6], bad reception contributes significantly to the overall energy consumed by the radio. Because a smartphone spends on average around 23 hours per day in idle state [7], this radio policy reduces the majority of bad reception side effects on the radio energy consumption.

Second, the proposed policy reduces the amount of unnecessary traffic produced by smartphones. On average, 19% of all the traffic generated by a smartphone is produced during the idle state [6]. Most of this idle state traffic can be considered unnecessary because it provides no immediate functionality to the user. Idle state network traffic is often the result of careless application design, such as not taking the extra steps to consider the smartphone's state before generating network traffic. But also, it is the result of smartphone operating systems not enforcing stricter resource allocation policies for the data interfaces. In Android and iOS, for example, applications do not require a special set of permission to use the data interfaces while the smartphone is in the idle state. Thus, it is easy for a developer to make an application that mistakenly continues to produce network traffic even while the smartphone is in the idle state.

It should be noted that the radio frugality policy will not be active when the smartphone is in active state. This policy is only used when the phone is in idle state.

B. SMS-based Push Service

In current smartphone world, push services use long-lived TCP connections to create "virtual circuits" from push servers to smartphones. Using these virtual circuits, push services are able to send data that the smartphones have not requested. However, to maintain the virtual circuits, the data network interfaces of smartphones must remain on at all times. Thus, disabling the data interfaces by our proposed scheme while a smartphone is in the idle state would prevent notifications from being delivered to the smartphone. This may not be acceptable for some useful real-time apps, such as email, social networking, or instant messages.

To allow smartphones to continue receiving notifications even while the data interfaces are disabled, we propose using SMS as a side channel to facilitate the delivery of notification data. Because SMS uses traditional cellular technology to route and deliver messages [16], it enables the push service to inform a smartphone of incoming notification even when the phone's data interfaces are disabled.

The first step for an application to receive notifications through the proposed SMS-based push service is to register the phone's number with the application server. This can be easily done when the app is installed and connects to the application server for the first time. The application server then stores the phone number for future use. Once the application server has new data for the app, it simply sends a wakeup SMS message to the smartphone using the phone number received during the registration process. The SMS message may contain an optional but small payload (e.g., the ID of the app, type of the notification message, or the complete notification data itself if it can be fit into one SMS message), or contains no payload at all. Upon receiving the SMS wakeup message, the SPS client code on the smartphone will enable the data interfaces (3G/4G cellular and/or Wi-Fi) to synchronize with the server using a data connection (or, if the SMS message payload contains all the necessary information, assimilate the message directly). Figure 2 illustrates the complete procedure of our proposed SPS scheme.

Besides allowing the data interfaces to be disabled while the smartphone is in idle state, the SMS push is inherently more energy efficient than typical push services on data channels. First, maintaining the virtual circuit between the smartphone and the server required by traditional push services introduces additional and unnecessary idle state traffic. It requires the smartphone to re-establish a long-lived TCP connection with the push service every time the smartphone loses and regains data connection, switches wireless data technology, or changes gateway. In addition, it requires periodic transmissions of keep-alive packets to prevent firewalls or NATS from dropping the long-lived TCP connection [11]. Because SMS uses traditional cellular technology to route and deliver messages, the proposed SMS push does not produce any unnecessary idle state traffic. Furthermore, SMS messages are delivered through cellular control channels, and as a result, the radio does not enter the high power state when receiving an SMS notification [17].

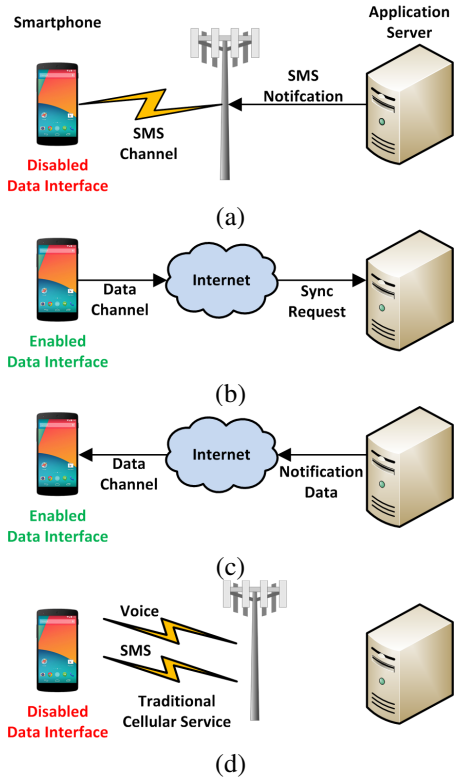

Fig. 2. SMS-based push service procedure for the proposed SPS in phone's idle state. (a) When the application server has new data for the smartphone while the phone's data interfaces are disabled, it first sends an SMS message to the smartphone. (b) The SMS notification prompts the smartphone to enable its data interfaces and send a sync request message to the application server using the regular data service. (c) The application server replies to the sync request by sending the new data to the smartphone. (d) Once the smartphone receives the new data, it disables its data interfaces again if it remains in idle state. The phone can still receive voice calls or SMS messages via traditional cellular service.

If a smartphone user wants to further save energy consumed in phone's idle state, or reduce the amount of SMS notification messages issued by SPS push server, the user can easily configure the SPS push server to deliver SMS notification messages in a more intelligent way, such as deliver each SMS message only after it accumulates up to n notification requests, or after such as 10 minutes has passed. By configuring the parameter to be infinity, the user can completely remove energy consumed by data interfaces during phone's idle state.

IV. Prototype and Evaluation

Currently, third-party app servers that want to push notifications to smartphones do so by using a public or private push service, such as "Google Cloud Messaging for Android" [9]. These services then deliver the notifications to the corresponding push service client running in the target smartphone by using a protocol such as SMPP or MQTT.

To measure the energy utilization of the proposed SPS scheme, we have developed an SPS push server and an Android client prototype. The source codes of the server and client used throughout the evaluation are available online at http://www.cs.ucf.edu/~czou/SPS.

A. Prototype

We implemented our prototype SPS push server using Java on an Amazon AWS server. As previously discussed, to deliver a sync notification the SPS server first sends an SMS message to the smartphone. These SMS messages are specially formatted for the SPS client running in the smartphone. Specifically, each message starts with the key phrase "Server Hello" and contains a universally unique identifier (UUID) [18]. The key phrase allows the SPS client running on a smartphone to distinguish SPS notifications from regular SMS messages sent to the user. Once the SPS client has detected an SPS notification, it retrieves the notification data from the SPS server by enabling the data interfaces of the smartphone and send the unique notification ID to the push server. The SPS push server uses this UUID to retrieve the appropriate message from a message queue and sends it back to the smartphone using the newly established data channel.

There are multiple methods to send SMS messages programmatically from a computer. In our prototype, we use the SMTP SMS gateways provided by cellular carriers mainly because they are free of charge. These gateways are able to convert SMTP messages to SMS messages. To use these gateways, our prototype push server simply sends an email to `phonenumber@carrierdomain.com`. The prototype server uses Oracle's JavaMail library [19] to connect with Gmail server and uses Gmail as a proxy to send an SMS message to one of these SMTP SMS gateways. We deploy this method in our prototype because it is cost free, although it is not the quickest delivering method.

In real deployment, third-party servers generate the notification traffic and then should use the SPS push service to deliver the message. For our experiments, the notification traffic is simulated by the prototype SPS server itself. The prototype server generates notification traffic that follows a Poisson process.

The SPS client is implemented as an Android app that automatically begins running in the background when a smartphone is powered on. The SPS client is responsible for both enforcing the proposed radio frugality policy and handling the SMS-based push service. The SPS client automatically disables or enables the data interfaces (both 3G/4G and Wi-Fi) every time the smartphone enters the idle or active state respectively. In addition, the SPS client monitors all incoming SMS messages looking for SPS notifications. If an SPS notification text message is received, the SPS client first deletes the text message and then begins the server sync process. To do this, the SPS client enables the data interfaces, retrieves the data from the application server using a TCP connection, and disables the data interfaces again. By discarding the SPS text message, the client prevents the SPS notification text message from reaching the Android text messaging application, and thus, making the SPS operation transparent to smartphone users.

Throughout our experiments the SPS push server runs in an Amazon AWS server and the SPS client runs in an LG MS840 Android smartphone with a 1,540 mAh battery. The LGMS840 smartphone is able to use 1xRTT CDMA, EVDO, and LTE network technologies throughout the experiments. In addition, we have implemented a test Android app and a test application server that uses the current Google's GCM push

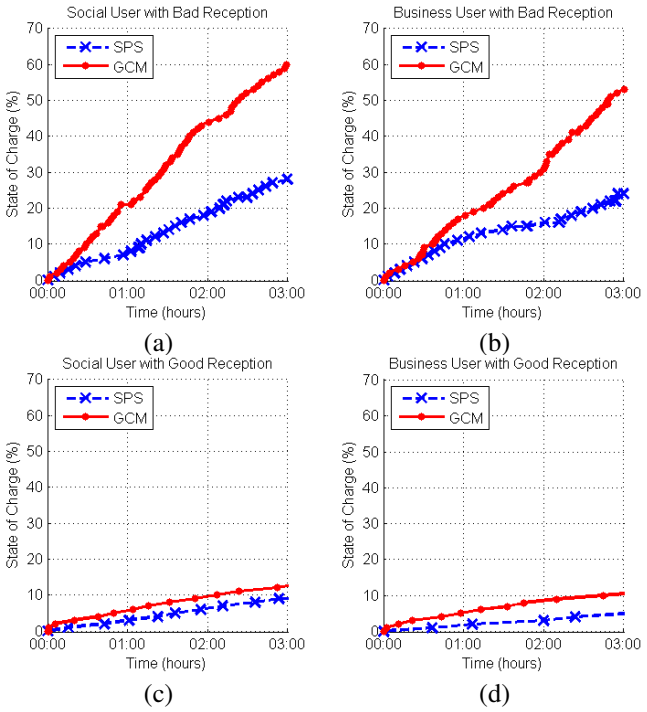

Fig. 3. Battery charge used by SPS vs GCM of (a) a social user, and (b) a business user, experiencing bad reception. Battery charge used by SPS vs GCM of (c) a social user, and (d) a business user, experiencing good reception.

service [9]. We use the GCM implementation to compare how the proposed SPS performs against traditional data channel based push services.

Finally, it is worth mentioning that although we only use sync notifications in all the experiments, our prototype does support data notifications purely using SMS channel without data channel involved. In this mode, as long as the notification data is not big, it can be directly encoded into one or several SMS messages sent to the smartphone, where the SPS client absorbs notification data without any further action. Such notification method would consume even less energy than SPS sync notifications because the data interfaces do not need to be enabled after receiving an SMS notification message.

B. Battery Usage

To measure the battery impact of the proposed scheme, we simulated notification traffic and use the SPS and the GCM implementations to handle it, respectively. In addition, we define two types of users to study how the number of notifications affects push service energy consumption. The first type, the social user, is defined as a user that actively uses Facebook, WhatsApp, email. The second type, the business user, is defined as a user that only needs real-time notifications for email service. Given that on average Facebook, Whatsapp, and email applications receive 82, 116, and 105 notifications throughout the day [20] [21] [22], we define the social user and business user notification traffic as on average 303 and 105 notifications per day respectively. We simulate the notification traffic using a Poisson process, and assume that all notifications are received within a 16-hour time window in each day.

We ran the social and business user notification traffic simulations for 3 hours using both the SPS and GCM implementations. In addition, we performed each experiment in an area with bad reception, where the signal strengths for 1xRTT/EVDO and LTE are on average -110 dBm and -130 dBm respectively, and also in an area with good reception, where the signal strength for 1xRTT/EVDO and LTE are on average -85 dBm and -94 dBm respectively. Throughout each experiment we record the battery's state of charge. The state of charge is calculated by the Android operating system to indicate how much of the battery's capacity has been used. Figure 3 shows the result.

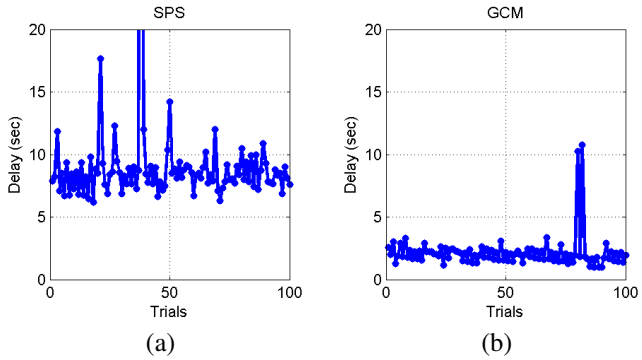

Fig. 4. The delay to push a notification using (a) SPS scheme, and (b) Google's GCM scheme.

Figure 3 shows that SPS consumes significantly less energy than GCM when the smartphone is in bad reception area. In fact, given our test environment, the result figures show that the battery of a smartphone using SPS would last twice as long as that of a smartphone using GCM.

Figure 3 shows that SPS also consumes less energy than GCM when the smartphone is in a good reception area. Comparing energy use between good reception and bad reception, we can conclude that bad reception is a major power drainage factor, and the proposed SPS works best if a smartphone stays a long period in bad reception areas throughout a day (which happens frequently for employees who daily work inside metal-framed buildings).

C. Delay

We measure the delay of delivering a notification using SPS against GCM. The results are shown in Figure 4. It takes on average 9.8 sec to deliver a notification using our prototyped SPS system. On the other hand, it takes on average 2.1 sec to deliver notification using GCM.

We further analyze the time delay of SPS notifications by recording each event that occurs on the prototype server and Android app. To ensure that our measurements are accurate, we synchronize the smartphone and the push server using a `pool.ntp.org` time server [23]. We can split the SMS push notification delay into three parts:

- The SPS push server sends an email to the email proxy;
- The email proxy sends SMS message to the smartphone;
- The smartphone activates data interfaces to retrieve notification data from the SPS push server.

Figure 5 shows the time delay for the three parts in SPS notification. On average, 1.9 sec are spent in sending the notification to the proxy email server used in our cost-free

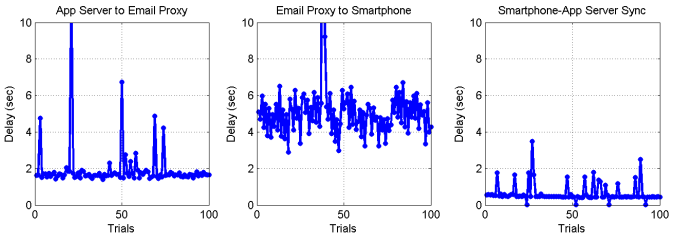

Fig. 5. Time delay in SPS scheme. (a) Delay between the application server and the email proxy. (b) Delay between the email proxy and the smartphone. (c) Delay to retrieve the new data from the application server.

implementation; and 6.2 sec is spent by the cost-free email proxy to complete SMS message delivery. We used the SMTP SMS gateway method in our prototype because it is provided by the cellular carriers free of cost. There are alternative methods to send SMS notifications without involving an SMTP gateway, which could reduce the SPS delay significantly by removing the email proxy from the notification transmission procedure.

In addition, we observe that the extra step of enabling the data interfaces before retrieving the notification data adds minimal delay. On average, it takes 630 msec for SPS to sync with the application server while GCM takes 487 msec.

V. Conclusion

In this paper we propose a new energy saving scheme to reduce unnecessary power consumption when smartphones are in the idle state, especially when they are in bad reception areas. First, our scheme implements a policy that disables the data interfaces every time the smartphone enters the idle state and re-enables it every time the smartphone enters the active state. By doing this, the proposed policy is able to neutralize two of the main sources of energy consumption related to the data interfaces: bad reception and unnecessary idle state traffic. To support real-time notification functionality required by some apps, we propose a notification system that relies on SMS to temporarily enables the data interfaces to retrieve notification data through normal data channel. This SMS-based method allows smartphones to continue to receive notifications even when the data interfaces are disabled.

We developed a prototype of the proposed energy saving scheme and tested in a realistic environment. Our evaluation shows that SPS consumes significantly less energy than the current approach. In fact, using the proposed SPS the battery of a smartphone lasts twice longer than using the current system when the smartphone is experiencing bad reception.

References

[1] J. Heggestuen. (2013) One in every 5 people in the world own a smartphone, one in every 17 own a tablet. [Online]. Available: http://www.businessinsider.com/smartphone-and-tablet-penetration-2013-10

[2] Free apps drain smartphone energy on 'advertising modules'. [Online]. Available: http://www.purdue.edu/newsroom/research/2012/120404HuSmartphone.html

[3] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "Characterizing radio resource allocation for 3g networks," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 137–150.

[4] J. Huang, F. Qian, A. Gerber, Z. M. Mao, S. Sen, and O. Spatscheck, "A close examination of performance and power characteristics of 4g lte networks," in *Proceedings of the 10th international conference on Mobile systems, applications, and services*. ACM, 2012, pp. 225–238.

[5] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*. ACM, 2010, pp. 281–287.

[6] N. Ding, D. Wagner, X. Chen, Y. C. Hu, and A. Rice, "Characterizing and modeling the impact of wireless signal strength on smartphone battery drain," in *Proceedings of the ACM SIGMETRICS/international conference on Measurement and modeling of computer systems*. ACM, 2013, pp. 29–40.

[7] J. Fetto. (2013) Americans spend 58 minutes a day on their smartphones. [Online]. Available: http://www.experian.com/blogs/marketing-forward/2013/05/28/americans-spend-58-minutes-a-day-on-their-smartphones/?WT.srch=PR_EMS_smartphones_052813_press

[8] C. Miller, D. Blazakis, D. DaiZovi, S. Esser, V. Iozzo, and R.-P. Weinmann, *IOS Hacker's Handbook*. John Wiley & Sons, 2012.

[9] Google cloud messaging for android. [Online]. Available: http://developer.android.com/google/gcm/index.html

[10] X. Zhang and K. G. Shin, "E-mili: energy-minimizing idle listening in wireless networks," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 9, pp. 1441–1454, 2012.

[11] H. Haverinen, J. Siren, and P. Eronen, "Energy consumption of always-on applications in wcdma networks," in *Vehicular Technology Conference, 2007. VTC2007-Spring. IEEE 65th*. IEEE, 2007, pp. 964–968.

[12] M. Gupta, S. C. Jha, A. T. Koc, and R. Vannithamby, "Energy impact of emerging mobile internet applications on lte networks: issues and solutions," *Communications Magazine, IEEE*, vol. 51, no. 2, pp. 90–97, 2013.

[13] L.-S. Meng, D.-s. Shiu, P.-C. Yeh, K.-C. Chen, and H.-Y. Lo, "Low power consumption solutions for mobile instant messaging," *Mobile Computing, IEEE Transactions on*, vol. 11, no. 6, pp. 896–904, 2012.

[14] P. C. Dinh and S. Boonkrong, "The comparison of impacts to android phone battery between polling data and pushing data," in *IISRO Multi-Conferences Proceeding. Thailand*, 2013.

[15] A. Pathak, Y. C. Hu, and M. Zhang, "Bootstrapping energy debugging on smartphones: a first look at energy bugs in mobile devices," in *Proceedings of the 10th ACM Workshop on Hot Topics in Networks*. ACM, 2011, p. 5.

[16] K. Amri and T. Ceglarek. Sms: How does it work? [Online]. Available: http://services.eng.uts.edu.au/userpages/kumbes/public_html/ra/sms/

[17] M. Sauter, *3G, 4G and Beyond: Bringing Networks, Devices and the Web Together*. John Wiley & Sons, 2012.

[18] A universally unique identifier (uuid) urn namespace. [Online]. Available: http://www.ietf.org/rfc/rfc4122.txt

[19] Javamail. [Online]. Available: http://www.oracle.com/technetwork/java/javamail/index.html

[20] The average facebook user receives 82 notifications everyday. [Online]. Available: https://twitter.com/UberFacts/status/263487232613163008

[21] S. McGlaun. Whatsapp handles 50 billion messages daily, more than sms delivery. [Online]. Available: http://www.slashgear.com/whatsapp-handles-50-billion-messages-daily-more-than-sms-delivery-21313892/

[22] Q. Hoang. Email statistics report, 2011-2015. [Online]. Available: http://www.radicati.com/wp/wp-content/uploads/2011/05/Email-Statistics-Report-2011-2015-Executive-Summary.pdf

[23] How do i use pool.ntp.org? [Online]. Available: http://www.pool.ntp.org/en/use.html