Staying Online While Mobile: The Hidden Costs

Andrius Aucinas
University of Cambridge

Narseo Vallina-Rodriguez
University of Cambridge

Yan Grunenberger
Telefonica Research

Vijay Erramilli
Telefonica Research

Konstantina Papagiannaki
Telefonica Research

Jon Crowcroft
University of Cambridge

David Wetherall
University of Washington

ABSTRACT

Mobile phones in the 3G/4G era enable us to stay connected not only to the voice network, but also to online services like social networks. In this paper, we study the energy and network costs of mobile applications that provide continuous online presence (e.g. WhatsApp, Facebook, Skype). By combining measurements taken on the mobile and the cellular access network, we reveal a detailed picture of the mechanisms selected to implement online presence, along with their effect on handset energy consumption and network signaling traffic. We are surprised to find that simply having idle online presence apps on a mobile (that maintain connectivity in the background, with no user interaction) can drain the handset battery nine times more quickly. This high cost is partly due to online presence apps that are excessively "chatty", in particular when their design philosophy stems from a similar desktop version. However, we also find that the cost of background app traffic is disproportionately large because of cross-layer interactions in which the traffic unintentionally triggers the promotion of cellular network states. Our experiments show that both of these effects can be overcome with careful implementation. We posit that a two-way push notification system, with messages being sent at a low (regular) frequency and low volume by a network-aware sender, can alleviate many of the costs.

Categories and Subject Descriptors

C.2.3 [**Computer-communication networks**]: Network OperationsNetwork Monitoring

General Terms

Design, Measurement, Performance

Keywords

Energy, Mobile, Radio, Cellular, Networks, RNC

1. INTRODUCTION

The emergence of smartphones over the last few years has provided a new platform for application developers to develop and showcase innovative apps. However, connectedness for mobile applications (apps) means being connected to the cellular network as well as some Internet service for the delivery of real-time information (e.g., a message or voice). We refer to this kind of connectedness as *online presence*. Apps that feature online presence are among the most popular apps, and seem likely to increase in popularity in the years to come. A remarkable example is WhatsApp, a service that handles 10B messages per day[1].

Energy is at a premium and radio is one of the most power-hungry components on mobile devices. As online presence uses the radio and incurs costs even when the user is not engaged with the device, it is important that apps implement online presence connectivity as efficiently as possible. For cellular voice connectivity, progress over the past two decades has led to highly efficient implementations on 3G/4G: high-end devices today offer a nominal lifetime of up to 12 days on a single battery charge while continuously connected to the network but idle. Unfortunately, recent work suggests that online presence connectivity is much more costly: periodic transfers and traffic sent while the screen is off can consume most of the radio energy even though it is a small fraction of the overall traffic because it triggers cellular network state transitions [1].

In this paper, we explore the implementation of online presence in popular applications and the associated energy and network costs in detail. We advance on prior work [1–3] by studying online presence and by reporting on network signaling costs measured over the cellular interface. Unlike prior work, which uses transport-layer packet traces to infer costs and summarize behavior, our work runs controlled experiments on an instrumented mobile device, that accurately records application and radio state information in addition to network packets, and in a cellular handset compliance facility. The collected measurements give us an accurate and detailed view of the app traffic, network signaling, and energy consumption of online presence for a set of popular apps.

The main contribution of this paper is the results of our experimental study. We find a diversity of implementations

[1]`http://news.yahoo.com/whatsapp-now-delivers-10-billion-messages-day-175531856.html`

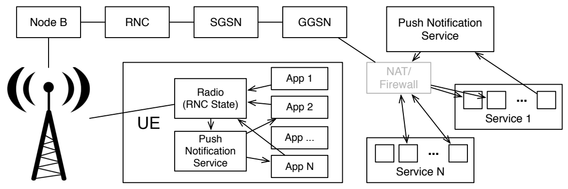


Figure 1: Interaction between applications and online presence services over a cellular network.

of online presence. Some apps implement online presence with a minimal network and energy footprint that involves very little signaling. Other apps, particularly those with equivalent desktop versions, tend to be "chatty" and have high network signaling overheads; porting apps from the desktop to the mobile is not a good strategy. As a consequence, the power consumption of online presence is very poor compared to cellular connectivity: running popular apps in the background increases the battery drain by up to a factor of *9* even when no work is performed.

To maintain online presence, developers need to deal with platform constraints and differences (such as maintaining a connection to the service even when in background on iOS and Android - Sec. 2) and network constraints (such as the presence of middleboxes like NATs [4] with different timeouts [5, 6] for which state should persist), as well as characteristics of different networks [5, 7].

Our work informs the discussion on the best way to structure mobile apps. We believe that the most efficient way to ensure energy efficient online presence is through the design of a platform API that will allow application developers transparent continuous connectivity without detailed understanding of the fundamentals behind cellular operation. However, we find the platform-native push mechanisms that already exist to be limited. We propose that future push notification systems provision for bi-directional transfers, at low frequency and low volume to avoid unnecessary network signaling and energy waste. Architectural changes exposing network state may also help to achieve spectrum-friendly operation.

2. BACKGROUND

The focus of our work is online presence applications on mobile phones. The fundamental difference between online presence applications and other apps on phones is that the former need to maintain connectivity with the server offering the service, even when they run in the background, for the timely delivery of incoming messages to the phone. There are a number of issues that need to be addressed for such continuous support.

2.1 The Role of the Platform

The general operation of online presence is illustrated in Fig. 1: User Equipment (mobile device) runs multiple applications, some of which use the platform's push notifications or manage their own communications. Communications traverse the cellular network to the Internet, where remote services and push notification services are hosted.

Both in iOS and in Android, an application moved to the background can continue its network operations until it is ultimately killed either by memory constraints or a timeout of the platform. Background execution on mobile devices has been identified as problematic in the past[2]. For these reasons, both Apple and Android have introduced native push notification services[3]. Hence, when an app is put in background or killed, it can use the push notification, a *one-way* mechanism that enables 3rd party servers to send data to the phone. Both implementations (Android and iOS) use XMPP-like mechanisms to implement push notifications. Importantly, when an application is alive it can use any mechanism.

The advantage of a push API is that all applications requiring such support can communicate with the phone using a single platform specific API, thus utilizing a unique connection of the device to the platform support servers. The limitation of the push API is that it only allows communication from the support server to the mobile, and we increasingly see applications needing to also communicate with the server at different points in time and while in background. Furthermore, for open platforms like Android, the use of the push API is merely recommended; there is *no* enforcement policy in place (like in iOS). Finally, some particular implementations (iOS) may impose further constraints in its use, such as explicit service declaration (VoIP, Audio streaming, Accessory support, Bluetooth...) because the platform prioritizes these types of applications, finite-length tasking, and eventually memory consumption limitation. Due to the all of the above, some applications choose to develop their own connectivity logic, i) either to enable the transfer of information from the client to the server, or ii) to avoid any constraints imposed on the push API use (see Sec. 4).

2.2 The Role of the Network

The maintenance of online presence typically relies on some kind of interaction between the phone and the online service, that directly relates to energy expenditure for the handset. To ensure the freshness of the online presence information, network activity has to happen at frequent intervals. Moreover, given that the mobile device is likely to be behind a number of middleboxes in the cellular network (like NATs), such activity needs to occur at a rate faster than the time it takes for state to expire on those middleboxes (something that is highly network dependent). If applications stay in the background for long periods of time, like in Android, they are bound to generate a number of such connections just to maintain connectivity to the service.

2.3 The Role of the Radio Technology

If the traffic generated by the online presence apps in background is the only traffic sent by the device, then it is also very likely to be the one responsible for power state transition for the cellular modem. Given that all our experiments have been carried out on 3G networks (LTE is not available in the countries we performed experiments), we will use 3G terminology for the rest of the paper. The fundamental concepts are similar in LTE.

In the 3G RNC (Radio Network Controller) state machine, if the mobile device sends no traffic, it remains on IDLE

[2]Apple stated background execution as a battery killer while introducing 3rd party app support in iOS2 (June 2007), `http://www.theverge.com/2011/12/13/2612736/ios-history-iphone-ipad`

[3]Apple introduced it in iOS3, fully supported in iOS4. Android followed similarly with the service named Google Cloud Messaging.

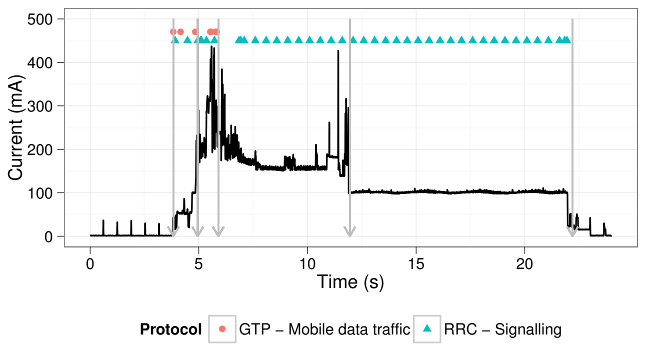


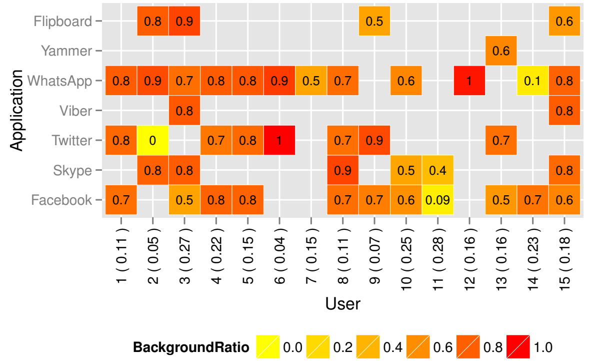
Figure 2: UMTS current drain trace correlated with signaling event from the test facility.

power state with minimal energy expenditure and network resource allocation. When the device sends/receives traffic it moves from IDLE to FACH (3G), a higher power state that allows the device to access a low speed, shared communication channel. A dedicated communication channel (DCH for 3G) is assigned to the mobile after its stay in FACH for a certain period of time or after the transmission of a certain amount of volume (network specific) [5]. The release of resources is timer based [6]. Then depending on the network generation, the resources might be completely released (return to IDLE state, where the IP connectivity is dropped), or still allocated (PCH state, where the mobile can still be paged to recover its connectivity in a fast way).

While the energy consumption related to power state transitions has been the focus of recent research, less is known about the associated signaling cost. To understand the true effect, we measured the signaling and energy cost of data transmissions on a fully equipped network testbed provided by a large cellular equipment vendor. The testbed consisted of a complete radio access network, RNC and SGSN as in Fig. 1. Logging of packets is done at multiple points: RNC, SGSN and GGSN, as well as the mobile device. This equipment is connected through the GGSN to the core network of a major European operator to provide handset compliance testing. We also use a Monsoon power monitor [8] to track the power consumption of the devices we study.

Fig. 2 provides a complete view of the signaling traffic, and electric current drawn during a sample data transmission event for one of the apps. To the best of our knowledge, this is the first such study of signaling traffic for apps. At $t = 4$, an inbound packet from the Internet triggers a signaling event (paging) that powers up the radio of the phone, first in FACH (shared channel, low power), and then immediately in DCH (dedicated channel, high power). The promotion from FACH to DCH happens if the carried volume exceeds 128 Bytes, as configured in the network. This parameter as well as the idle timers that determine the precise shape of the example trace in the figure vary across different networks [9] and consequently, the cost of communications can be slightly different, as we analyze later.

Note that signaling overhead is not only introduced during state transitions, but also throughout the entire phase that the mobile device is in high power state (small triangles between $t = 6$ to $t = 22$), right after data transmission and during tail time in DCH. The reason behind this behavior is that the mobile device needs to send measurement reports every 0.5-1 seconds to inform the RNC of the channel quality, in order for the RNC to trigger handover, adjust modulation and channel coding rate, and take power



Application	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Flipboard		0.8	0.9						0.5						0.6
Yammer													0.6		
WhatsApp	0.8	0.9	0.7	0.8	0.8	0.9	0.5	0.7		0.6		1		0.1	0.8
Viber			0.8												0.8
Twitter	0.8	0		0.7	0.8	1		0.7	0.9				0.7		
Skype		0.8	0.8					0.9		0.5	0.4				0.8
Facebook	0.7		0.5	0.8	0.8			0.7	0.7	0.6	0.09		0.5	0.7	0.6

Figure 3: Ratio of background vs. total running time for 7 most used applications across 15 users. Fraction of time the user had the display on for in parentheses.

control decisions. The release of the DCH channel has been traditionally thought to happen after a network configured timeout $T1$ [10]. In this network setup, we found out that the duration of the timeout depends on the actual data rate over the channel, further complicating the picture.

From this experiment, we note: i) DCH is very intensive in terms of resource usage and thus inappropriate for the transmission of small bursts of data. ii) Promotions from one state to the other are accompanied by a number of signaling messages. We measure those to be *5* for each PCH to FACH promotion; *10* for each PCH to DCH (with 1 message per measurement if needed), and at least *13* for each IDLE to DCH (and 1 message per measurement if needed) transition. Although there is a direct mapping from RNC transitions to signaling messages and hence the number of RNC transitions conveys the amount of signaling, the large number of messages also explains why short transmissions are so expensive and have high latency. iii) FACH consumes less energy and causes less signaling in addition to keeping the radio active for shorter period of time by not having to wait for DCH promotion and demotion first. Thus, our measurements indicate that if online presence is used only for "keepalive", it should minimize the number of promotions and, ideally, be supported in FACH.

3. ONLINE PRESENCE MAINTENANCE

The primary goal of this work is to analyze the sources of the high online presence costs. In this section we look at the different online presence methods employed by applications and their overheads.

3.1 The Apps

Online presence applications are those that require some kind of network connectivity to Internet services, even when they run in the background. We recruited 15 Android users to identify the apps with such characteristics and their typical usage. We ask the users to install a monitoring tool on their handsets that collects i) the power state of the 3G modem, ii) if the display is on or off, iii) the applications running in the foreground (if the display is on), and iv) the applications in the background (display on or off), every minute. The study was carried out during the entire month of Dec 2012. The main results are shown in Fig. 3.

Fig. 3 shows the seven applications with the longest running times in addition to Google services, that are running all the time on Android handsets. Among them, Facebook,

Application	Total	FACH	DCH
Facebook	34.4	20.5	13.9
Google Apps	2.3	1.4	0.9
WhatsApp	1.0	0.3	0.7
Skype	41.3	18.0	23.3
Viber	21.0	18.6	2.4

Table 1: Percentage of RNC promotions with traffic of a particular app.

Skype, Viber, WhatsApp are popular *online presence* apps - apps that maintain constant connectivity and allow the user to be contacted by other connected individuals. All online presence apps tend to run in the background most of the time, 70-80%, and they are among the most popular apps across different platforms – App Store (Apple) and Google Play (Android) as of June 2013. In Fig. 3, we also report the percentage of time that the users have the display on throughout our study. This is important to capture given the high energy requirements of the display itself. We find that users actively interact with the handset, rather infrequently, e.g. only 16% of the time on average, and as little as 4%. Therefore, background network activity could significantly contribute to overall energy expenditure by the handset.

3.2 The Mechanics

To understand how applications maintain online presence we modified a popular Android device, the Samsung Galaxy SII device to frequently poll the status of the radio layer on a per second basis. The polling interval was limited by how frequently the radio firmware responds to requests, but this granularity was sufficient for our analysis. In parallel, we intercept all traffic transmitted over the 3G radio and associate it with the generating process. The tool we used for radio information and traffic logging is described in detail in previous work [9]. This way, we can attribute packets to processes that generated them over time and the specific 3G power mode the packets were sent on. Looking at our collected traces in detail, less than 0.2% of the packets from very short-lived connections, present in 8 out of the 3577 total promotions, are missed. We verify the results of our tool against detailed traffic traces obtained from the network testbed and verify their accuracy. For this analysis, the basic inactivity timeouts $T1$ and $T2$ are both 6 seconds for the operator under study. Fast Dormancy is disabled on this network.

We focus on the five most popular online presence apps identified above, e.g. Facebook, Viber, Skype, WhatsApp, and google services. We setup a Google account and installed each one of the online presence apps we found in our user study, one at a time after a new fresh install. We use a dummy user with a single contact to sign into different services. The single contact makes no attempt to send a message to or call our test user.

We run each app in isolation for a period of 3 days in the background on a European 3G network. We observe varying levels of radio use between the different apps. Table 1 shows the proportion of promotions that contained traffic from a particular application. The total number of promotions during a 3-day period was 3577. We further break the promotions down to ones that only went to the FACH state and ones that reached DCH. The table points towards very different online presence strategies between the applications: Google Applications (including Google Cloud Messaging) and WhatsApp are present in a very small proportion of total promotions, which is not the case for the other three apps. Furthermore, on this network Viber's traffic primarily causes promotions to FACH channel with lower energy footprint, while traffic of Skype can be found in as many as 41.3% of all promotions, out of which 23.3% were to DCH, suggesting that it is very taxing to the battery of a mobile device.

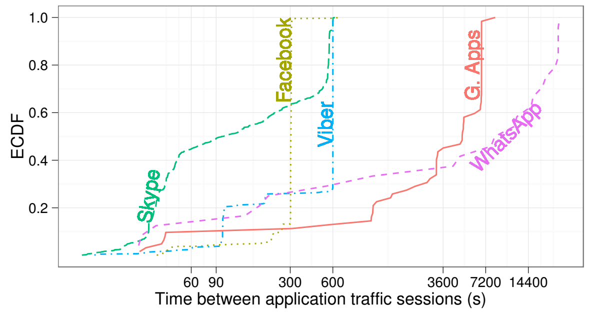


Figure 4: Empirical CDF of the time between application keepalives.

In Table 2, we observe peculiar patterns for each application that already indicates a potential lack of common practices in mobile application development. Skype is a standout example in terms of its radio use, low periodicity and the number of different hosts connected. We observe it uses a set of super-nodes to traverse NATs and middleboxes [4] to establish direct connectivity between peers over UDP and TCP, over multiple ports including standard ones 80, 443. This indicates a strategy to evade network firewall restrictions. We also find that Skype contacts more than 500 hosts, with a very high frequency (nearly once every 240s, as indicated on Fig. 4).

Viber uses a similar P2P scheme, but with a limited numbers of nodes and with an optimized network payload, as the amount of DCH-based promotions is comparatively low (2.4%). It mostly uses non-standard port (TCP 4244), with a lower median periodicity of 600s. We find that Facebook seems to be relying on a client/server architecture, largely distributed (47 hosts contacted over 3 days). Facebook contributes to a large number of promotions (20% and 13.9%, to FACH and DCH respectively) with a low and steady frequency of once every 300 seconds on multiple ports (80, 443, 8883). Looking at the hostnames Facebook is contacting, we identify different services, and most notably that port 8883 seems to be responsible for the Facebook Chat service.

On the opposite end of the spectrum, Google and WhatsApp are exhibiting optimized network access, with around 1% of total promotions in either mode. The period between messages from these applications is quite long, on average 4000s and 10000s respectively. However, Fig. 4 also indicates that the period is not only long, but also dynamically adapted over a wide range of values - we speculate that the developers use adaptation algorithms depending on the network environment. Once again, the traffic is on ports 5222 and 5228 that are designated as XMPP-based services but the traffic is SSL-based and we could not verify the exact nature of the messages.

RNC transitions also depend on the amount of data transferred. To this extent, we look at the distribution of data volume per promotion for each app. The distribution of traffic exchanged per application is shown as a Violin plot (Fig. 5) separated into client and server- originated transfers. We

Application	Hosts	Ports		Periodicity (s)			Signaling (est.)	
		UDP	TCP	Mean	Median	95%	FACH	DCH
Facebook	47	-	80, 443, 8883	287.0	303.2	306.0	3658	7937
Google Apps	12	-	80, 443, 5228, 7276	4166.6	4911.3	6726.3	249	513
WhatsApp	23	-	443, 5222	10342.9	8350.4	23404.0	53	399
*Skype	585	(*)	80, 443, (*)	239.6	91.9	598.8	3212	13305
*Viber	15	5243, 9785	80, 4244, 5242	470.1	600.0	600.3	3319	1370

Table 2: Number of hosts connected, UDP and TCP ports used, and periodicity between the RNC promotions. Signaling is estimated based on the number of messages per promotion. (* - Viber and Skype use direct connections between peers and potentially any port.)

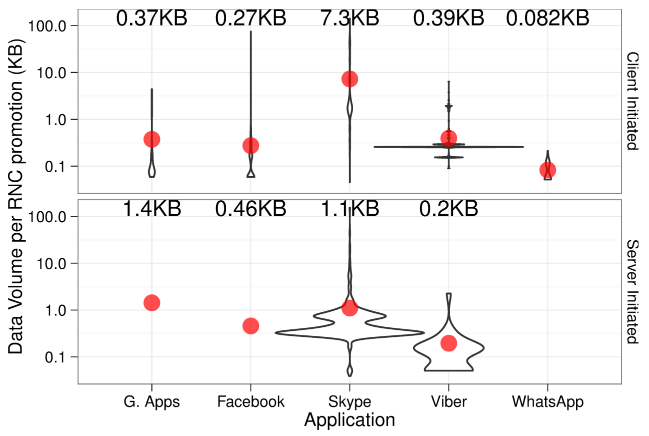


Figure 5: Distribution of traffic generated per RNC promotion to maintain online presence for each app. The text label indicates the average volume per promotion.

RNC	Power	% of Time	Time at a State (s)	
			Mean	95 Percentile
DCH	100u	6.8	9.3	9.5
FACH	40u	9.6	4.7	4.8
PCH	<2u	83.6	62.2	63.9
IDLE	1u	0.0	0.0	0.0

Table 3: Time spent at a given RNC state (user with a single contact) and theoretical power for each mode in arbitrary unit as reported by the GSM Alliance.

note that most of the traffic is client-based, with the exception of the P2P apps that also initiate server-to client connections. As expected, the amount of traffic transmitted in a large fraction of promotions is less than 100 bytes! The majority of the traffic is from the client to the server, a scenario that is natively not supported through the platform-specific push API. What is more, Viber and downlink communications of Skype send a very similar amount of data in each promotion, as indicated by the distribution in the Violin plot, while others are more varied.

Regarding signaling, our results show that Facebook and Skype are large contributors, due to their use of a dedicated channel (DCH) and its high cost to establish and maintain online presence. Viber, even though optimized for FACH access, still suffers from the chatty P2P nature. Google and WhatsApp are the only apps that minimize the signaling load efficiently, even though all Google services use the same channel.

3.3 Energy Overhead

Although online presence applications run in the background most of the time, they can contribute to a large portion of device energy consumption. In Table 3, we map the observed RNC states to the potential energy consumed for the experiment we ran in the previous section. Even if high energy states such as FACH and DCH are used less than 10% of time, they might represent a substantial loss as the power level observed might reach 40X and 100X the level of the energy spent in idle.

To quantify their energy overhead, we connected the same phone to a Monsoon power monitor. We measured the amount of energy consumed by the device idle without apps and with different combinations of the apps installed. We run the tests on two different networks to account for the effects different RNC state machine configurations may have (Table 4[4]).

We found out that just by installing and authenticating one of the online presence services, we may drain the handset battery four to five times faster. The cost is very high due to mobile traffic required to maintain applications active even without any interaction! When we measure the current drawn when multiple online presence applications are running in parallel, the battery lifetime can decrease by a little over a factor of 9 over the case with no applications – the device stays on for barely over one day on a single charge just because of having these applications installed and running in background. Even though this is an extreme case, it is common to see the device lasting for little over 2 days with only idle online presence, compared against over 10 days battery life of a completely idle device. The cost becomes particularly high considering the relatively short proportion of time such applications stay in foreground (Fig. 3).

4. TOWARDS BEST PRACTICES

Our characterization points to a number of design implications. First, mobile applications need to be developed specifically for the cellular/mobile scenario, by understanding and incorporating the unique characteristics of cellular networks: radio states, timeouts and energy costs. Applications such as Facebook and Skype that communicate with their infrastructure often and exchange large messages cause significant energy waste. For example, while P2P is inexpensive on a desktop equipment and has benefits for the content provider, it becomes very costly on mobile devices and networks. Techniques such as dynamic keepalive period adaptation on the other hand can have very positive effects as demonstrated by WhatsApp and Google services.

Second, that platform APIs need to be more versatile to satisfy the needs of applications. In particular, an optimized uplink channel is necessary - every one of our tested applications regularly initiates communication with the remote servers. This pattern also suggests that the main purpose of this traffic is to ensure *connectivity* rather than receive new information. The necessity for such traffic is unclear, specially considering the presence of platform Push API calls,

[4]Current drain (mA) is chosen over power (mW) as power depends on voltage that decays with the battery discharge

Applications	**Network 1** T1=6s, T2=6s			**Network 2** T1=8s, T2=12s		
	Average current (mA)	**Projected battery life (h)**	**Drain speed**	**Average current (mA)**	**Projected battery life (h)**	**Drain speed**
None	6.1	268.7	**x1**	5.9	279.7	**x1**
Google services	9.0	183.7	x1.5	22.5	73.3	x3.8
Google, WhatsApp	12.3	134.4	x2.0	28.3	58.4	x4.8
Google, Viber	12.6	131.4	x2.0	27.5	59.9	x4.7
Google, Skype	17.2	95.9	x2.8	31.8	51.9	x5.4
Google, Facebook	10.2	162.6	x1.7	22.9	72.1	x3.9
Google, Skype, WhatsApp, Viber	22.4	73.5	x3.6	54.5	30.3	x9.2

Table 4: Idle current drawn of online presence applications, projected battery life, and speed of battery drain compared to idle, on networks N1 and N2.

but a possible rationale is that these APIs (either Google or Apple) do not support two-way communication and usually provide no delivery guarantees. However, Google recently announced they will support two-way communications in their push API[5]. This might not address all problems if not strictly enforced, a difficult choice considering that even Apple had to change their position about background applications and have recently started talking about opening more of their APIs [6].

Third, we find that applications tend to transmit varying amount of data to maintain online presence. While this suggests that the messages go beyond simple keepalives, such messages tend to be small in volume, and by nature latency insensitive. We posit that online presence should be separated from other functionality to further reduce periodic message size. Furthermore, given the increasing trend of online presence embedded in mobile apps, one could consider throttling push related traffic to a rate that would not cause a promotion to DCH, in the absence of other traffic.

Overall, it is necessary to take low-level network management details into account when designing applications and their background activity patterns, requiring developers to become more "spectrum-aware". Unfortunately we do not think that the complexity of cellular networks can be easily explained to the new wave of mobile developers, especially developers used to the web platforms, working alone or in small teams and having little prior experience with cellular networks. The solution has to come from the platforms, and the mobile OS has to decide whether specific data is useful for the user and use network resources appropriately. Research should therefore explore *platform-centric* approaches, while understanding user-behavior to tackle these problems.

5. CONCLUSIONS

Most popular mobile apps stay connected to an Internet service over and above the connectivity between the mobile and the cellular network. We study the costs of this online presence in depth. To do so, we collect data on the activity of 15 users, identify the most popular background applications, and then analyze their traffic patterns correlated with their induced network state activity. We base our analysis on our observations in a cellular test facility, combined with low level information extracted from the device. We find that mobile apps have diverse strategies for online presence that are not well met with a simple push API, which results in an excessive cost for online presence, in terms of signaling and battery life. Simply maintaining online presence with no user interface activity increases the power drain by up to 9 times compared with no online presence. Our characterization argues for a platform-based optimization, offering dual way communication and with more respect to the cellular network constraints.

[5]`http://developer.android.com/google/gcm/ccs.html`
[6]`http://techcrunch.com/2013/05/29/meet-tim-cooks-apple-apis-abound-spending-money-to-make-money-long-bets-on-new-categories/`

Acknowledgments

We would like to thank our reviewers and our shepherd Fabián E. Bustamante. We would also like to thank the network engineers who helped us at the mobile operator testing facility.

6. REFERENCES

[1] 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 *WWW '12*.

[2] J. Huang, F. Qian, Z. Mao, S. Sen, and O. Spatscheck. Screen-off traffic characterization and optimization in 3g/4g networks. In *IMC '12*.

[3] P. Athivarapu, R. Bhagwan, S. Guha, V. Navda, R. Ramjee, D. Arora, V. Padmanabhan, and G. Varghese. Radiojockey: mining program execution to optimize cellular radio usage. In *Mobicom '12*.

[4] Z. Wang, Z. Qian, W. Xu, Z. Mao, and M. Zhang. An untold story of middleboxes in cellular networks. In *SIGCOMM'11*.

[5] F. Qian, Z. Wang, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. Characterizing radio resource allocation for 3G networks. In *IMC'10*.

[6] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani. Energy consumption in mobile phones: a measurement study and implications for network applications. In *IMC'09*.

[7] J. Huang, F. Qian, A. Gerber, Z. Mao, S. Sen, and O. Spatscheck. A close examination of performance and power characteristics of 4G LTE networks. In *MobiSys'12*.

[8] Monsoon Power Monitor. `http://www.msoon.com/LabEquipment/PowerMonitor/`.

[9] N. Vallina-Rodriguez, A. Aucinas, M. Almeida, Y. Grunenberger, K. Papagiannaki, and J. Crowcroft. Rilanalyzer: a comprehensive 3g monitor on your phone. In *IMC '13*.

[10] R. Huoviala and A. Pihlajamäki. Optimised messaging patterns, October 13 2008. US Patent App. 12/734,303.