Daniel Burgstahler, Nils Richerzhagen, Frank Englert, Ronny Hans and Ralf Steinmetz: Switching Push and Pull: An Energy Efficient Notification Approach. In: Proceedings of the 3rd International Conference on Mobile Services (MS 2014), p. 1-8, June 2014.

Switching Push and Pull: An Energy Efficient Notification Approach

Daniel Burgstahler, Nils Richerzhagen, Frank Englert, Ronny Hans and Ralf Steinmetz
Multimedia Communications Lab (KOM), Technische Universität Darmstadt, Darmstadt, Germany
Email: {firstName.lastName}@KOM.tu-darmstadt.de

***Abstract*—An increasing number of modern smartphone applications are dependent on information updates from the cloud. To realize such information updates mainly two communication approaches are common, namely *push*- and *pull*. Due to different communication patterns both approaches differ in their energy consumption and notification latency. The energy constrained nature of mobile devices entails a sensible selection of the appropriate notification approach. In this paper we provide an evaluation of the energy consumption of both communication approaches. Based on this we provide a transition approach that is able to use the best of both, low latency and low energy consumption. Our results show that energy savings of up to 7 % of the total smartphone battery per day can be achieved by switching between both approaches, depending on the context.**

***Keywords*—client notification, push, pull, energy efficiency, energy consumption**

I. Introduction

Mobile phones have rapidly evolved in the last two decades from large devices only capable for simple phone calls towards inventive smartphones. Todays devices offer a lot of services, including mobile Internet connection, geographic positioning and innovative multimodal user interfaces. The virtually unlimited possibilities of these devices have fostered the rapidly increasing popularity of smartphones and led them become ubiquitous devices in everybody's Pocket. The number of smartphones in use is continuously increasing and has already reached 1.4 billion devices [1]. Just last year the number of sold smartphones even excelled 900 million devices [2]. This increasing popularity combined with the growing amount of built-in functionalities fosters attractiveness to develop novel apps and services and the concept of easy-to-install apps allows rapid integration.

Due to the mobile nature of smartphones they are battery powered and thus, energy is a limited resource. However, while the computing capabilities of smartphones are increasing to roughly twice as much every two years and thus, corresponding to Moore's law, the development of batteries is significantly slower. The energy density of batteries "did not even double over the last decade" [3]. Thus, the only way to increase the battery capacity is to increase the size of the battery but the size is determined by the design of the smartphone. Therefore, energy saving strategies become very important. The amount of available energy is fixed and has to be used carefully by applications to prevent draining out the battery to fast.

The energy consumption of mobile devices strongly differs for different operating modes. An operating mode with a high energy consumption is the wireless transmission of data that is also the base for many upcoming application scenarios. This is pushed by the rapidly increasing amount of external data sources that can be used as a foundation for new application concepts. An example of such data sources and application scenarios is the rapidly increasing intelligence of cities and buildings, caused by the deployment of sensors. Many cities are already deploying a lot of sensors to become part of a smart environment that is able to provide near real time information with high granularity about it's current status of traffic, parking situation or air pollution. Traffic flow can be monitored on the level of streets or even intersections and available parking space can be observed up to a level of single parking spaces. On the one hand large research projects push the deployment of city sensors, e. g., in the city of Santander, Spain, where 12,000 sensing devices have already been deployed, triggered by the research project SmartSantander [4], [5]. On the other hand cities start to prove possibilities of sensor deployments by already commercially available systems, e. g., the city of London that hat started to deploy 3,000 parking sensors [6]. Also the the area of buildings and homes is getting more and more equipped with sensors to become smart. An example for this clear trend is that Google recently invested more than three billion US dollar in a home sensor company [7]. Another application example is the real time monitoring of machine and transportation status in cross-organizational manufacturing processes [8]. The used mobile sensing devices have to transmit status changes immediately and operate energy efficiently [9].

All these new information sources provide the foundation for new information services that are the enabler for value-added mobility services. These new services are able to inform end users about changes in traffic situations and regulations, provide congestion warnings, information about free parking spaces and dynamic routing. The design of necessary information infrastructures and how service platforms can integrate the variety of different information sources is also a current focus in research projects [10].

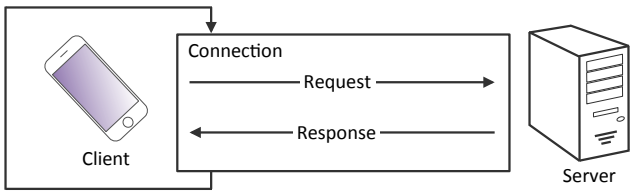

Figure 1. Schematic illustration of the operating principle of *pull*-based notifications of mobile clients.

An overlap in many modern application scenarios, such as the smart city scenario outlined before, is the notification of mobile clients of new information. Due to the relatively high energy consumption of wireless communications these information updates should be optimized with respect to energy consumption to increase the operating time of the smartphone [3]. For the notification about information updates in smartphones two different paradigms exist, namely *push*- and *pull*- based notification. Both have different characteristics in their energy demand and latency. In our previous work [11] we have already shown that an educated choice of the notification paradigm can save a lot of energy. *Pull* based approaches are able to save energy compared with *push* based approached but results in higher latencies. However, in many scenarios the latency of information updates has the highest priority. Hence, the research question we aim to empirically answer in this work is: "How to design a novel notification paradigm that exploits high energy efficiency and low latency in parallel?"

In the next Section II, we introduce the experimental design of our energy measurements, followed by the presentation and a discussion of our measurement results in Section III. Based on these results we introduce our novel transition based notification approach in Section IV and derive possible energy savings within an analytical evaluation in Section V. We give an overview of related work in Section VI. In the following Section VII we give some considerations about privacy and finally conclude our paper with a summary and outlook in Section VIII.

II. Experimental Design

In this section the overall design of our energy measurements is described. We begin with a brief explanation of the considered dependent and independent variables. In the following we describe the used notification approaches and their respective implementations. This is followed by a description of the used measurement tool and the measurement procedure.

A. Considered Variables

The focus of this work is estimating costs of different client notification approaches and provide sensible concepts to decide which of the former should be used. We understand the cost of these notifications as the energy they demand and therefore, this is our only *dependent variable*. We consider, as in our previous work, the average power consumption to build on previous results and to maintain comparability between already conducted experiments. The average power calculates from the demanded energy over time divided by the execution time.

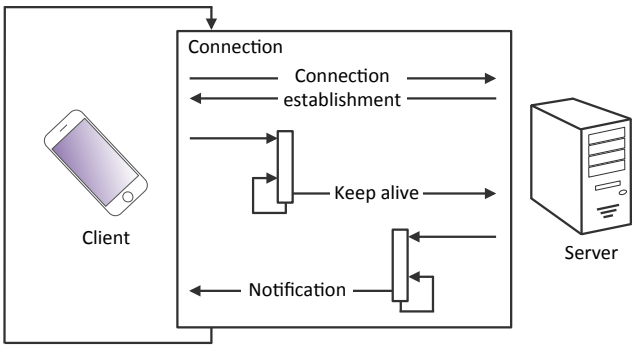

Figure 2. Schematic illustration of the operating principle of *push*-based notifications of mobile clients.

A set of *independent variables* may influences our dependent variable, i. e. energy consumption. This set consists of adapted parameters that we identified as important factors for the energy demand in notification technologies in our previous work [11] and new parameters that we consider as potential factors for energy demand. As the packet payload has not a significant impact on the energy-wise outcome [11] we considered a payload size of 100 Bytes for our experiments. A message interval of 1200 s was chosen by us to compare our own implementations of the notification approaches, which are presented later in this section. We considered 1200s as message interval as with this the impact of the TCP connection timeout messages rises. At lower messages intervals, like 180 s, connection timeout messages (keep alive) would for example not occur. In this paper we also consider the impact of encryption techniques, *Transport Layer Security* (TLS) and multiple implementations of push and pull mechanisms leading to our later presented architectural concept.

B. Notification Approaches

With respect to the notification paradigm we distinguish between a *push*-based and a *pull*-based invocation pattern. As implementation platform for our notification experiments we have used Google Android [12] for the mobile client.

1) PULL: Using a pull based notification pattern a central server buffers new information that is received from external sources. The client, in our case a smartphone application, periodically establishes a new connection to this server and requests for information updates. After the information exchange the connection is closed and reestablished at the beginning of the next interval. A schematic illustration of this approach is given in figure 1. On the server side we have used a Apache Tomcat[1] server with a RESTful interface that provides information as JSON data. As encryption for the communication between our server and the smartphone we have used TLS, similar to *Google Cloud Messaging* (GCM) which was used for our *push*-based application which is explained in the following.

2) GCM-based PUSH: In contrast to this, by using a *push*-based invocation pattern the connection to the central server is established once by the client and kept open. To realize this it is common to use a TCP socket with longer timeouts. To prevent the connection to be closed by intermediate network components, e. g. routers may drop unused connection entries after an internal timeout, the client device has to transmit periodically messages to the server. These messages are commonly named *keep-alive* packets. Since in this approach the server has a direct connection to the client, new obtained information updates can be directly transmitted to the client without the need of buffering the data. Thus, the client is able to receive information updates without additional latency. A schematic illustration of the *push*-based approach is given in Figure 2. For the implementation of our *push*-based application we have used the previously mentioned GCM[2]. Using GCM the Android smartphone maintains a connection to the Google

[1] http://tomcat.apache.org

[2] http://developer.android.com/google/gcm/

servers. Information updates from the application servers are not directly send to the mobile client. Instead this data is sent to the Google servers and forwarded to the mobile client. In Apples iOS[3] and Microsofts Windows Phone[4] similar notification implementations are used.

3) Push.KOM: In addition to the encrypted GCM *push* implementation we have implemented an own *push* service with a corresponding Android application to compare the impact of encryption. The underlying communication is TCP based with a keep-alive interval of 180 s. As indicated by Choi et al. [13] *keep-alive* intervals in common implementations fluctuate between 30 s and 10 minutes. Our *keep-alive* interval was arbitrary selected and has empirically shown to result in a stable connection.

C. Measurement System

Smartphones are a comparatively closed system that makes it difficult to measure the energy consumption of single tasks with a high accuracy. To do so one can follow several approaches. Using a software tool might be the simplest way. In the respective application market stores several energy measurement applications are available. All these solutions are based on the same functional principle. The execution time of single components is measured and the energy consumption is then estimated based on an energy model. A similar but advanced software solution is PowerTutor [14] that is able to determine a device specific energy model automatically. However, since all these solutions are based on estimations also the results are not comparable with precise measurements. To get a precise result one has to use an accurate external measurement device. The sampling frequency must be high enough to recognize even very fast changes in energy consumption, e. g., the transmission of short messages.

Nevertheless a direct external power supply is not possible for modern smartphones since the devices communicate with their batteries. Because of this the device will not start up if the feedback from the battery is missing. Therefore, one has to power the device with the original battery to get realistic measurements. To be able to plug our measurement device in between the battery and the smartphone we used a similar setup as proposed in [15]. We used a modified charging cradle to plug in the original battery and be able to easily contact wires to the battery. The charging cradle ha no other function than to connect the battery, all insides were removed. To connect to the smartphone we produced a dummy battery. Just a plastic body with the same size and shape as the original battery. This dummy battery has electrical contacts at the same position as the original battery. The wires from the charging cradle are connected to these contacts. The original battery is plugged into the charging cradle and the dummy battery into the smartphone. Thus, we are now able to intercept the power connection between the original battery and the smartphone. In between we placed our power measurement device. A schematic illustration of the electric circuit is given in Figure 3

To measure the energy consumption we have used a Hitex Powerscale [16] combined with an *Active Current Measurement*

[3]https://developer.apple.com/library/ios/documentation/NetworkingInternet/Conceptual/RemoteNotificationsPG/Chapters/ApplePushService.html

[4]http://msdn.microsoft.com/en-us/library/hh221549.aspx

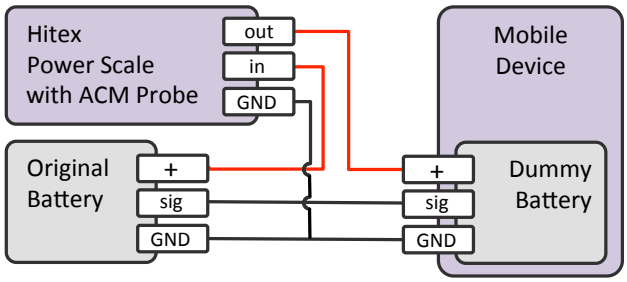

Figure 3. Schematic illustration of the electric circuit of the measurement system, with ground (GND), voltage (+), and signal (sig) lines. The latter serves a serial communication between the smartphone and the battery to read charging status and battery temperature.

(ACM) probe. Additional measuring resistors would distort the results. The use of the ACM technology makes such additional measuring resistors dispensable. The Power Scale measurement device has a high sampling resolution of up to 100 kHz, is able to measure in the range of nano ampere and has a low error in measurement of only 1 % for voltage metering and 2 % for current measurement according to the manufacturer.

D. Measurement Procedure

For our measurement we conducted several combinations of values for the three independent variables. However, in this paper we focus only on a very limited subset because in this work our emphasis is to derive a new transition based notification paradigm. A more detailed view on the different variable permutations we have already provided in our previous work [11]. Our comparison is on a constant update interval of 1200 s and a notification payload of 100 Bytes. We have measured this for both notification paradigm, *push* and *pull*. For *push* we have used Google Cloud Messaging that encrypts data by the use of TLS and additionally an own implementation that does not use any encryption. For the *pull* approach we have used a Tomcat Server with a RESTful interface. Here we have also used TLS for encryption. We have used a sampling rate of 100 kHz to measure the power consumption. Each run had a duration of 90 minutes and was repeated five times, leading to 2.7 billion samples per configuration of parameters.

We have used WiFi for the data connection. A dedicated access point was deployed and placed in direct proximity to the smartphone to reduce external interference. The wireless channel was set to an not used one in the surrounding and there were no other WiFi devices within the same room. To get comparable results we stopped all possible background processes on the smartphone and started our measurement after the smartphone switched into power save mode, only executing our application in background. Also all vibration, optical, or acoustical notifications were switched off and received notifications were only written to a log file.

III. Experimental Results and Discussion

In our consideration we focused on the difference in energy consumption of the different notification approaches. The payload size was set to a fixed value of 100 Bytes, while maintaining a constant notification interval of 1200 s. We have executed our experiments for both notification approaches.

Figure 4. Results of the average energy consumption for the three different notification techniques of mobile clients.

Our *pull* implementation and the GCM solution both encrypt all communication based on TLS. For comparison we have also used an own *push* implementation that does not use any encryption. In our previous work we have already shown that for small update intervals, e. g., 180 s, all considered notification approaches have a similar average power consumption and for longer update intervals the difference is significant [11]. However, in this work the shown results are a foundation for the derivation of our new notification paradigm, described in the next section. Thus, we focus here on the results for larger message update intervals that are given in Table I and for an easy comparison also in Figure 4. It can be clearly seen that for larger message update intervals, i. e., 1200 s, the average power consumption strongly differs depending on the notification paradigm. The higher average power consumption for the *push*-based approaches can be explained by the overhead caused by maintaining the connection to the server. In this specific example the average power consumption of the *pull* approach is about 19 % lower than for the GCM based *push* approach. The results also indicate that the use of an encrypted connection causes a higher energy consumption that can bee seen in the difference of the two *push* approaches. However, these two are not directly comparable since both are based on different implementations.

Our results show that for a static notification interval the *pull* approach may exhibit a much lower average power consumption. For the *pull*-based approach the communication is directly conducted between the application server and the mobile device which is an advantage in case of small message update intervals because it has no external server in between. However, the use of a *push* based mechanism provides a clear advantage of lower latency in message delivery. Messages can be directly transmitted to the mobile client when they arrive at the messaging server since the communication channel is kept open by the mobile client at all times. At the end it strongly depends on the application scenario which notification mechanism is best suitable. For exclusive externally determined application

Table I. COMPARISON OF THE MEAN POWER CONSUMPTION FOR ALL CONSIDERED RUNS (SAMPLE SIZE $n = 5$).

Paradigm	Run Payload Size	Interval	Avg. Power [mW]
Pull	100 Bytes	1200 s	91.47
GCM-based PUSH	100 Bytes	1200 s	112.88
Push.KOM	100 Bytes	1200 s	103.73

cases, e. g., instant messaging services like WhatsApp[5], the *push* mechanism has clear advantages because of the lower latency. The use of a *pull* based mechanism would lead to large delays in message delivery in such a scenario or alternatively a large number of continuous client requests would be necessary that would cause an increase of the energy consumption.

The impact of the difference can be illustrated by the following example that is related to our ongoing work in the European Union-sponsored SIMPLI-CITY project [10]. A user starts a journey to a destination within a big city and needs a free parking lot next to the desired destination. The sophisticated navigation App executed on the mobile device is able to retrieve information about free parking lots in the destination area. If in such a scenario the App makes use of a *push* mechanism, it would register for information updates of the target area at the beginning of the journey. The connection has to be maintained open and the according information updates will be received continuous, immediately when they arise. In contrast to this, by the use of a *pull* mechanism the application does not need a connection to the server side until the user is next to a parking area and can then start a dedicated request. In this example the application is able to determine relatively exactly the point in time when an information update is needed. In this example, the App can decide the need of an information update depending on the current context, i. e., the current position. Since information updates are only necessary in a relatively short time interval in contrast to the whole journey, here the *pull* mechanism has the potential to reduce the energy demand of the mobile device.

IV. TRANSITION APPROACH

The previously presented results have shown that both notification paradigms have a significant difference in energy consumption. However, based on this knowledge one can not determine that one of the paradigms is the preferable one since both have specific benefits and drawbacks. In summary it can be said that the choice of the best notification paradigm is context dependent. Since the application context can change and in many cases the context is determinable by the App executed on the mobile device, a next step is to switch between the available notification paradigms depending on which approach fits best to the current context. Thus, we introduce our concept of transition between *push* and *pull* at runtime.

Our approach is based on the assumption that the mobile client is able to determine the current context. We define the context as the probability p that an information update δi would be necessary within a timespan t_1 that is smaller than the regular information update interval t_{update}. The interval t_{update} describes the *pull* information request interval which is equivalent to the maximum information latency. If the probability $\bar{p} = 1 - p$ is high that there is no need for a

[5] http://www.whatsapp.com

Table II. COMPARISON OF THE BATTERY CAPACITY OF SOME LATEST SMARTPHONES.

Smartphone	Battery Capacity [mAh]
Galaxy S3 [17]	2100
Galaxy S4 [17]	2600
Nexus 5 [18]	2300
Desire 700 [19]	2100

faster information update then we can use the *pull* approach that is more energy efficient in this case. However, if the context determines the probability p for the need of an information update is high, then we switch to the *push* approach to benefit from the low latency in information updates. The threshold X for the selection of the notification approach can be determined by the application. The standard value is $X = 0.5$ An illustration of the principle of this approach is given in Figure 5. In the case the system had decided to use the *pull* approach we use a timer with the interval t_{update} for the next information request. After the *pull* information request is performed we restart the decision of the appropriate notification approach. In case of the use of the *push* approach we set also a timer with the interval t_{update} to restart the decision of the appropriate notification approach. The application can continuously determine the probability of the need of an information update based on the current context. In both cases the decision of the appropriate notification approach is executed periodically in an interval of t_{update} due to the previously named timers. The value of t_{update} can be determined by the application to get the best equilibrium of notification latency and energy conservation.

To realize a transition based approach, the notification server has to buffer all new information and has to implement an interface that enables the client to request for information updates as *pull*. Additionally this application server has to provide the *push* functionality. If the *push* or *pull* approach is used is determined by the client. The client is also responsible to establish the connection to the application server and maintain the connection open in case of the *push* approach. If a new information arrives at the application server, this tries to deliver this information as TCP packet to the last known address of the mobile client. If the server receives an acknowledgement from the mobile client, the information data can be deleted from the server. If no acknowledgement is received, the information data has to be buffered until the client reconnects to the server to request for an information update.

V. ANALYTICAL EVALUATION

Based on our previous description of the functional principle of our transition based notification approach we will analytical evaluate in this section the possible energy conservation of this approach. We will motivate and explain our analytical evaluation with the following story, which might be familiar for one or another. Anna, a former student who applied for a job after her masters, has today a job interview for which she has to drive from Pittsburgh to Downtown Manhattan. To find parking lots wherever Anna is going, she has a background service running on her smartphone which obtains free nearby parking places in real-time.

We now split the story in two parts regarding the notification approach the application is using. First, Anna's application

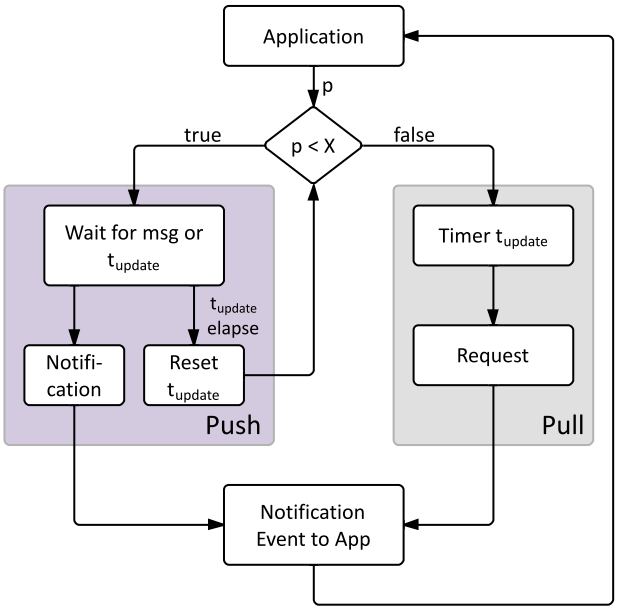

Figure 5. Architectural concept of our notification approach based on a transition between *push* and *pull* by switching both mechanisms according to the probability of an information update.

is completely push-based, running with the Google Cloud Messaging service. Thus, over 24 hours a day the background service keeps the push connection to the GCM server alive. This only push-based approach is as shown in the beginning of Section III and in Figure 4 expensive regarding the respective energy costs. One might say here that the application then should use a pull-based notification approach. Using such with an adequate pull interval of, e. g., 20 minutes to not overload the communication interfaces would result in no real-time update anymore, when Anna is Manhattan searching for a parking lot.

This signals us that the communication could be optimized. Therefore, we present the second part in which Anna's application is using our presented architecture from the former Section IV. Motivated by the fact that the search for a parking lot takes at most 30 minutes, which is already a very high value according to Thompson et al. [20], Anna's application would now use for 98% of the time, i. e., 23.5 hours a day, the energy saving pull notification approach. Only when her context gets more important, respective Anna arriving at Manhattan searching for a parking lot, the application switches from pull to push-based notifications for the last 2%, 30 minutes, of that day. By this, the application ensures in first place the freshness of the parking lot data by using the push notification approach when the data has to be delivered in real-time in case of an update, which can't be ensured by pull based notifications. Second, the application is able to reduce the energy consumption of the device compared to the before mentioned completely push-based application, again without loosing any accuracy or freshness of the retrieved data.

We now will compare both approaches energy-wise to demonstrate potential energy savings of our presented architecture. Equation (1) shows the energy demand of 'the only push-based' approach according to our findings in Section IV.

Running 24 hours the push-based notification scheme results in 2709 mW energy consumption.

$$24\,h \cdot 112.88\,mW = 2709.12\,mWh \quad (1)$$

Compared to the approach based on our architecture, which switches between pull and push notifications and uses for 23.5 hours the less demanding pull approach and only when needed (approximated 30 minutes a day) the real-time update providing push approach, we see in Equation (2) that it consumes 2206 mW.

$$23.5\,h \cdot 91.47\,mW + 0.5\,h \cdot 112.88\,mW = 2205.99\,mWh \quad (2)$$

Approximately 20 % less energy is consumed by our presented approach, compared to the state-of-the-art GCM push based approach.

$$\frac{2709.12\,mWh - 2205.99\,mWh}{3.7\,V \cdot 2100mAh} = 6.48\% \quad (3)$$

We set this in comparison to todays smartphone batteries that have around 2100 mAh, with a voltage of 3.7 V resulting in 7.77 Wh, for normal size smartphones and only sparsely more for larger smartphones. An overview of the battery capacity of some latest smartphones is given in Table II. Thus the possible energy savings per day by using our transition based approach, corrspond according to Equation (3) to about 7 % of the total battery capacity of a modern smartphone. This already results with only one application using a separate notification implementation. Considering the slow development of batteries that 'did not even double over the last decade' [3] this outcome becomes even more significant.

VI. Related Work

Today's way of using mobile devices for a variety of different tasks and being online all time causes a high energy demand. In contrast to this the energy density, and thus, the energy capacity, of smartphone batteries has only slightly increased over the last decade [3], [21]. Thus energy consumption has a higher importance than ever. Although energy is the most limited resource in mobile devices, the impact of *push*- and *pull*-based notifications has not gained much attention so far.

According to Balasubramanian et al. [22] a linear correlation between the used amount of data and the consumed energy for the wireless medium insists. Using the on device application *Nokia Energy Profiler*, thus falsifying the real smartphone energy consumption, the authors are able to show the direct proportionality of the transferred data and consumed energy. But sampling the average power by intervals of 4Hz does not capture fast changes of the operating mode or even the transmission of small data packets. With the measurement hardware used in our experiments we were able to sample with a 25,000 times higher resolution. An improvement to this is given by the approach of Yoon et al. [23]. The approach, named *AppScope*, directly monitors the hardware usage at kernel level. This precise information about the hardware usage gives an increasing accuracy of the estimated energy consumption.

Zhao et al. [24] examine the usage of push notifications for command dissemination in mobile botnets. To maintain the connection between the mobile client and the notification server the authors also investigate the heartbeat traffic. However, no precise energy measurements are performed, only an estimation of energy consumption based on software models is conducted.

802.11 network beacons for maintenance of connections, which are very similar to keep alive messages in *push* implementations, are examined respective their influence on energy consumption by Krashinsky et al. [25]. Concentrating on short beacon intervals, they are able to show the increase of energy consumption whose origin lays in disturbed sleep intervals of the devices. A related approach by Sharma et al. [26] shows similar results.

Using the aforementioned Nokia Energy Profiler, Perrucci et al. [3] estimate the energy consumption of different built-in hardware components of a mobile device. The authors also provide a comparison of energy consumption of communication components in their respective possible states.

Hasenfratz et al. [27] have analyzed the energy consumption of push and pull approaches for data collection in wireless sensor networks. According to their results, a pull based approach with dedicated data collection intervals can save a significant amount of energy compared to a push approach if long latencies can be tolerated. They recommend to use *pull* instead of *push* in data collection scenarios to save energy.

Hao et al. [28] presented eCalc, an application to estimate the CPU usage with respect to energy consumption. This software based estimation technique makes use of program analysis of mobile applications, also known as bytecode profiling, to obtain the energy demand of CPU usage. Still lacking the estimate of the overall power consumption or the power consumption of other components to address the energy-wise efficiency of different applications, which is not only dependent on CPU usage.

Choi et al. [13] study the impact of mobile Internet applications on the signaling traffic and thus, the impact on the overall mobile network traffic that mobile network operators have to handle. The authors highlight that applications force the radio resource frequently to change between connected and idle states, which in return causes the signaling overhead. While Choi et al. focus on the impact of the signaling traffic (e. g. by keep alive messages) from a network providers view, our approach focuses on solutions to overcome/reduce the battery drain of keep alive messaging, formerly push notification approaches, on the mobile device itself.

The opportunities of smart applications and the importance of context detection is analyzed by Lach and Mueller in [29]. They highlight the capability of dynamic Bayesian networks to model dynamic systems like the current context of smartphone applications. Their focus on context detection is the improvement of user needs that is similar to our strategy to improve the operating duration of the user's smartphone.

Papageorgiou et al. [30] analyze the energy demand of web service invocations on mobile clients as QoS parameter. They indicate the relevance of service invocation patterns and service call sequences with respect to the energy demand to enable energy-aware mobile applications that are based on web services.

In a first series of measurements we were able to show that energy savings of up to 19 % are possible by an educated choice

of the notification paradigm [11]. In contrast to this we focus here in more detail on the impact of the different notification paradigms – *push* and *pull* and derived an architectural concept of a transition based notification approach that is energy efficient and able to use the benefits of both notification paradigms. In summary, to the best of our knowledge, our work is the first that considers a transition based notification approach for mobile devices. Our results enable us to minimize the energy consumption caused by the notification component of an application and concurrently use the most opportune notification paradigm depending of the current application context.

VII. Privacy Considerations

Notifying the user about newsworthy content raises not only energy consumption concerns. It is also important to discuss privacy implications caused by different notification approaches. In this section we will briefly discuss those privacy considerations.

Vendor specific push services like Googles GCM rely on centralized push servers. All push messages send to the mobile clients flow over these servers. There is no end-to-end encryption applied, thus the push server can access the content of push messages. This allows the smartphone vendors to track the volume, frequency, destination and even the content of all push messages. Even without sensitive data in the push messages, parameters like the market share or usage trends can be derived from those data streams. At some point in the future, smartphone vendors might use this information somehow to increase their profit. Such unfair practices might affect different Application vendors rather than single end users.

If such a centralized setup is inappropriate for specific scenarios, the application vendor can switch to self hosted notification services. Then the security of the whole system is under full control of the Application developer. However, such a setup will increase the overall energy consumption, because another information channel must be maintained. In case of using a push service, a second persistent TCP connection must be kept alive and in case of a pull based solution, a second pull looper must be started to fetch new content. This increases the overall energy consumption, because the vendor specific push service could not be replaced with such a custom solution.

In summary, our findings clearly show the important impact of client notifications with respect to energy consumption. The best approach depends on the specific application and a trade off between latency, energy consumption and privacy requirements must be made.

VIII. Conclusions

The extensive availability of wireless high-speed data connections has massively changed functionality of mobile devices. Applications make use of data services and can provide information updates in near real-time. This enables advanced functionalities and increases the value for end users, e. g., by the dynamic use of new information to modify journeys and guide people intelligently through cities. But the energy consumption of end-user's mobile devices is influenced by the necessary information updates that have to be transmitted to realize such functionality. Therefore, developers have to decide about an appropriate notification paradigm to minimize battery drain.

In this work, our focus was on the introduction of our transition based notification approach. Our findings show that a hybrid push/pull approach with transitions between both operating modes could reduce the energy consumption depending on the application scenario by up to 7 % of the total battery capacity per day. This increases the battery lifetime of smartphones significantly. Nevertheless, the question arises how we can further reduce the energy consumption of mobile information systems. As shown in previous works, the energy consumption as well as the latency of the pull based approach depends on the duty cycle of the pull requests. If the client would know the exact availability of new information, the client could schedule pull requests in an optimal way. Unfortunately, an oracle predicting the arrival time of new information is not available in general. On the other hand, the energy consumption of the push approach is mainly caused by maintaining a persistent TCP connection with a push notification server. Thus, the energy consumption could be reduced by increasing the time between two connection keep alive messages. Unfortunately, the keep alive has a low upper bound. This is caused by intermediate network devices that drop inactive connection entries after a certain time, e. g., NAT routers that are used to connect smartphones with the public Internet. As shown in first experiments, connection-less Push services could further reduce the energy consumption, because no persistent connection must be maintained. Unfortunately, such an implementation would not work in the todays Internet due to the missing end-to-end connectivity of mobile clients.

A further advantage with respect to energy savings could possibly be a selective encryption. Whereas operation system built-in *push* mechanisms encrypt all communication, a mobile device could easily switch the encryption mode depending on the context when using the *pull* mechanism, since the message exchange is initiated by the mobile device.

In our future work, we aim to extend our experiments through the consideration of measurements of the implementation of our transition based notification approach. We additionally strive to take a selective switch of payload encryption into account and analyze the effects of different connection technologies such as GSM, 3G, and LTE.

Acknowledgements

Parts of the research leading to these results have received funding from the European Union's Seventh Framework Programme (FP7/2007-2013) under grant agreement no. 318201.

References

[1] H. Leonard, "There Will Soon Be One Smartphone For Every Five People In The World," Feb 2013, online at http://www.businessinsider.com/15-billion-smartphones-in-the-world-22013-2.

[2] Statista.com, "Global smartphone sales to end users 2007-2013 | Statistic ," Feb 2014, online at http://www.statista.com/statistics/263437/global-smartphone-sales-to-end-users-since-2007/.

[3] G. P. Perrucci, F. Fitzek, and J. Widmer, "Survey on Energy Consumption Entities on the Smartphone Platform," in *Proceedings of the 73rd IEEE Vehicular Technology Conference*, 2011.

[4] Libelium, "Smart City Project in Santander Monitors Environmental Parameters and Parking Slots," Feb 2013, online at http://www.libelium.com/smart_santander_smart_parking/.

[5] L. Sanchez, J. Galache, V. Gutierrez, J. Hernandez, J. Bernat, A. Gluhak, and T. Garcia, "SmartSantander: The Meeting Point Between Future Internet Research and Experimentation and the Smart Cities," in *Proceedings of the Future Network Mobile Summit*, 2011.

[6] S. Curtis, "Smart parking app begins rollout in London's West End - Telegraph," Jan 2014, online at http://www.telegraph.co.uk/technology/news/10573651/Smart-parking-app-begins-rollout-in-Londons-West-End.html.

[7] R. Winkler and D. Wakabayashi, "Google to Buy Nest Labs for $3.2 Billion," Jan 2014, online at http://online.wsj.com/news/articles/SB10001424052702303595404579318952802236612.

[8] S. Schulte, D. Schuller, R. Steinmetz, and S. Abels, "Plug-and-Play Virtual Factories," *IEEE Internet Computing*, vol. 16, no. 5, 2012.

[9] S. Zöller, M. Wachtel, F. Knapp, and R. Steinmetz, "Going All the Way Detecting and Transmitting Events with Wireless Sensor Networks in Logistics," in *Proceedings of the 8th IEEE LCN International Workshop on Practical Issues in Building Sensor Network Applications (SenseApp)*, Oct 2013, pp. 39–47.

[10] SIMPLI-CITY Consortium, "SIMPLI-CITY The Road User Information System of the Future," Oct 2013, online at http://simpli-city.eu.

[11] D. Burgstahler, U. Lampe, N. Richerzhagen, and R. Steinmetz, "Push vs. Pull: An Energy Perspective," in *Proceedings of the 6th IEEE International Conference on Service Oriented Computing And Applications*. IEEE, 2013, pp. 1–4.

[12] Google, "Android Developers," Feb 2014, online at http://developer.android.com/index.html.

[13] Y. Choi, C. hyun Yoon, Y. sik Kim, S. W. Heo, and J. A. Silvester, "The Impact of Application Signaling Traffic on Public Land Mobile Networks," *Communications Magazine, IEEE*, vol. 52, no. 1, pp. 166–172, January 2014.

[14] L. Zhang, B. Tiwana, Z. Qian, Z. Wang, R. P. Dick, Z. M. Mao, and L. Yang, "Accurate Online Power Estimation and Automatic Battery Behavior Based Power Model Generation for Smartphones," in *Proceedings of the 8th IEEE/ACM/IFIP Int'l. Conf. on Hardware/Software Codesign and System Synthesis*, 2010.

[15] A. Rice and S. Hay, "Measuring Mobile Phone Energy Consumption for 802.11 Wireless Networking," *Pervasive Mobile Computing*, vol. 6, pp. 593–606, 2010.

[16] Hitex, "Energy Optimization: PowerScale," online at http://www.hitex.com/index.php?id=1937.

[17] Samsung, "Android OS Smartphones - SAMSUNG," Feb 2014, online at http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/.

[18] Google, "Nexus 5 - Google," Feb 2014, online at http://www.google.com/nexus/5/.

[19] HTC, "HTC Desire 700 Specs and Reviews | HTC Global," Feb 2014, online at http://www.htc.com/www/smartphones/htc-desire-700/.

[20] R. G. Thompson and A. J. Richardson, "A parking search model," *Transportation Research Part A: Policy and Practice*, vol. 32, no. 3, pp. 159 – 170, 1998.

[21] J. Paradiso and T. Starner, "Energy Scavenging for Mobile and Wireless Electronics," *Pervasive Computing, IEEE*, vol. 4, no. 1, pp. 18–27, 2005.

[22] N. Balasubramanian, A. Balasubramanian, and A. Venkataramani, "Energy Consumption in Mobile Phones: A Measurement Study and Implications for Network Applications," in *Proceedings of the 9th ACM SIGCOMM Conf. on Internet Measurement*, 2009.

[23] C. Yoon, D. Kim, W. Jung, C. Kang, and H. Cha, "AppScope: Application Energy Metering Framework for Android Smartphones using Kernel Activity Monitoring," in *Proceedings of the 2012 USENIX Annual Technical Conference (ATC)*, 2012, pp. 36–36.

[24] S. Zhao, P. P. C. Lee, J. C. S. Lui, X. Guan, X. Ma, and J. Tao, "Cloud-Based Push-Styled Mobile Botnets: A Case Study of Exploiting the Cloud to Device Messaging Service," in *Proceedings of the 28th Annual Computer Security Applications Conference*, 2012.

[25] R. Krashinsky and H. Balakrishnan, "Minimizing Energy for Wireless Web Access with Bounded Slowdown," *Wireless Networks*, vol. 11, pp. 135–148, 2005.

[26] A. Sharma, V. Navda, R. Ramjee, V. N. Padmanabhan, and E. M. Belding, "Cool-Tether: Energy Efficient On-The-Fly WiFi Hot-Spots Using Mobile Phones," in *Proceedings of the 5th Int'l. Conf. on Emerging Networking Experiments and Technologies*, 2009.

[27] D. Hasenfratz, A. Meier, M. Woehrle, M. Zimmerling, and L. Thiele, "If You Have Time, Save Energy With Pull," in *Proceedings of the 8th ACM Conf. on Embedded Networked Sensor Systems*, 2010.

[28] S. Hao, D. Li, W. Halfond, and R. Govindan, "Estimating Android Applications' CPU Energy Usage via Bytecode Profiling," in *Proceedings of the first International Workshop on Green and Sustainable Software (GREENS)*, 2012, pp. 1–7.

[29] P. Lach and H. A. Mueller, "Towards Smarter Task Applications," in *Proceedings of the 9th IEEE World Congress on Services (SERVICES)*, June 2013, pp. 141–146.

[30] A. Papageorgiou, U. Lampe, D. Schuller, R. Steinmetz, and A. Bamis, "Invoking Web Services Based on Energy Consumption Models," in *Proceedings of the First IEEE International Conference on Mobile Services (MS)*, June 2012, pp. 40–47.

All online references in this paper were last accessed and validated in February 2014.