A Pushing Approach for Data Synchronization in Cloud to Reduce Energy Consumption in Mobile Devices

Carvalho, S.A.L., [#1], Lima, R.N., [#2] and Silva-Filho, A.G.[#3]

[#]*Informatics Center of Federal University of Pernambuco*

[1]salc@cin.ufpe.br

[2]rnl@cin.ufpe.br

[3]agsf@cin.ufpe.br

***Abstract* — With the increase in smartphone sales and the use of clouds, data synchronization is an essential activity in the mobile applications that aim to keep data up-to-date with information from the application server. This paper proposes an analysis of the main approaches for data synchronization concerning energy, which are the polling and pushing techniques. The results show a substantial gain in terms of energy consumption for the pushing technique when compared to the polling technique in mobile applications. An experimental environment based on Arduino and INA219 that allows current and voltage acquisition of 138 samples per second was used. Results showed a significant increase of 187% in terms of energy consumption when the polling technique was applied, which was more advisable than the use of the polling technique when one or more requests in a forty minute range are not necessary to be made.**

***Keywords*— energy consumption, energy measurement, smartphone, polling, pushing.**

I. INTRODUCTION

While the computational power of processors doubles every two years, the capacity of batteries is doubled only every 10 years [21]. One way to make the battery last longer, is to make it larger. An alternative to this need for greater autonomy in the batteries is to develop techniques that reduce the consumption to prolong the use of applications.

From 2010 to 2011 [7], there was a 58% increase in global sales of smartphones. According to the International Data Corporation - IDC [8], the amount of smartphones saw an increase of approximately 39 % from 2012 to 2013. In the third quarter of 2012 186.2 million devices were delivered against 258.4 million in third quarter of 2013. The Samsung devices led sales with 31.4% of the market share in 2013 against 13.1% by Apple. IDC also predicted sales for 2014 of 1 billion and 204 million devices, among which 78.9% using the Android operating system and 14.9% using iOS [13]. A report of CISCO [4] predicts that smartphone traffic will be increased geometrically every year until 2015.

Devices that use Android [9] operating systems have picked up sharply, rising from 200,000 devices activated per day in 2010 to 1.5 million devices activated per day in 2013 and today it is the platform with more active devices [18].

Along with the growth of mobile devices, came the need for the provision of services for these devices, solutions that were scalable to solve this problem and the clouds allow that. Therefore, a new research area called Mobile Cloud Computing (MCC) was created, focused on cloud solutions for mobile devices [6]. Clouds provide mobile devices with some advantages such as scalability, security, economy in device storage, when used properly, capable of providing savings battery transferring heavy processing to the cloud and low cost compared to the implementation on dedicated servers.

Given to the significant increase in the number of smartphones, it was possible to identify that batteries are crucial in the usability of these devices. A device that discharges very fast frustrates the user, preventing from the maximum use of the features of the device. Assuming that the power of the batteries is not following the evolution of devices and it is an essential item on the device, techniques are required to reduce energy consumption to prolong the battery ensuring the performance, usability and desired functionalities.

This work proposes an analysis of energy consumption by comparing two main techniques, pushing and polling. These approaches are used in data synchronization between mobile applications and their servers in cloud to reduce energy consumption in mobile devices.

Technically, polling consists of a device that regularly requests a server, updates data and close connection. In the pushing technique the device sends a request and keeps this connection alive. Whenever there is data to be updated, the server sends this data automatically.

This paper is organized as follows: section 2 describes related work focusing on the percentage of gain in energy savings of various techniques; section 3 looks into the concepts that are relevant in order to understand the proposal of this work; section 4 shows the components used in the experimental environment; section 5 outlines the proposal of the project and the methodology; section 6 presents the results and discussions and section 7 draws the conclusion of the study.

II. RELATED WORKS

The work presented in [19] proposes a technique that performs switching in data transmission between Wi-Fi or 3G taking into account the signal quality, network speed and data size. Energy needed to perform the operation is calculated from a model calculation of energy consumption is also proposed by the authors. Then the transmission technology at runtime is selected. The authors achieved a reduction in energy consumption of 18% compared to the use of Wi-Fi only and of 30% when compared to the use of 3G only.

The work in [17] demonstrates an approach that uses DVFS (Dynamic Voltage and Frequency Scaling) to reduce energy consumption. The study consists of an empirical analysis using hardware measurement from recurring activities on smartphones, such as data transfer by 3G and Wi-Fi, phone call and the cell phone in idle state. The authors conclude that the lowest frequency of CPU does not always implies less power consumption. This result is due to the fact that lower frequency operates for longer time while the higher frequency operates for less time. Therefore, the authors conclude that frequency appropriateness depends on the type of activity. At last, concludes that by switching to the proper frequency power saving of 30% is obtained when compared to the profile performance.

Much of the processing and data transfer is done in screen off mode. This mode keeps the screen disabled but continues to perform processing. The work presented in [12] performs an analysis of how much energy is spent in off-screen mode and also in the on-screen mode. The impact of the use of Fast Dormancy technique is analysed in [10] and Batching [5]. The authors conclude that the off-screen payload of packets is smaller than the one of on-screen. On average, 58.55% of energy consumption corresponds to data traffic. Furthermore, applying the techniques together can lead to 60.19% of savings in energy consumption, 25.33% in signalling overhead and 30.59% in channel delay.

In [16] the authors explain the technique of Fast Dormancy. It's characterized by a 3G connection that has an uptime after the end of the last data transmission. This time is useful to more data to be sent without having to re-establish another connection. The Fast Dormancy consists of closing the connection immediately before the end of the standard time for closing the connection. When the data transmission is finalized, the connection is closed.

The work in [21] finds ways to provide energy savings using techniques known in literature as Fast Dormancy, as processing versus transmission and processing in memory. The technique discussed by the authors, called the separation of transmission from processing, is to disconnect before starting the data process. The technique reduces the active and unused connection time. The other technique, called memory processing, tries to eliminate some write operations on flash memory and performing all operations in RAM. With these techniques, the authors reach a reduction of 44.3% in energy consumption.

In [11] a comparison between the main approaches utilizing Pushing (C2DM, XMPP, Xtity and Urban Airship) is made. Three variables are considered in this work, which are: stability, response time and energy consumption. They identified that the XMPP approach reached the best stability and response time, but the highest energy consumption. The Urban Airship approach reached the lowest power consumption.

In [22] a comparison concerning energy in polling and pushing techniques was made. The authors discuss about the need for energy savings in smartphones and perform tests on the Android operating system. They created two applications, one with polling and another one with pushing. They used a fictitious scenario. PowerTutor, a software that estimates the energy consumption of the application, was used in the experiments. The authors stated that the technique of pushing is more energy efficient than the polling technique.

The works in [19], [17], [12] and [21] use known techniques from literature and improve them to achieve greater energy savings. The work in [22] is very similar to the one proposed in this paper. It differs in the use of measurement in software (or simulated), whereas this work uses the hardware measure that is more accurate. The results in [22] are based on only one experiment, not being statistical relevance. The work in [22] does not specify when is better to use polling and how much the pushing approach is better than the polling one, when our work fills these faults. In this present work, however, efficiency is quantified and when traced to both techniques.

III. BACKGROUND

The pushing technique needs to keep an HTTP long connection open and within a certain time, e.g. 15 min, so that the server can send the updates. When time expires, the server sends an HTTP 200 OK and it is necessary to re-establish the connection. The pushing only works on mobile networks, not on Wi-Fi [21]. Mobile networks are available in almost all locations and remain active even when the phone goes into idle state (sleep mode), however the Wi-Fi connection is closed when entering idle state.

Google Cloud Message (GCM) is the successor for Cloud to Device Messaging (C2DM). These technologies allow data to be sent from a server to mobile devices without application having to request data. The information is sent to synchronize the data server and the mobile device. The GCM is free regardless the amount of traffic and data size.

The GCM uses the pushing technique to update data and provides a framework for easy handling and a scalable infrastructure for data synchronization between server and mobile application. The GCM has several qualities when compared to a personal or corporate server used to synchronize data between different applications. GCM avoids the need for a personal online server with high availability and eliminates the need for fixed IP in personal servers. The personal server is used to send data to the GCM and from there to the mobile device.

IV. EXPERIMENTAL ENVIRONMENT

In order to measure the energy consumption of the applications on the mobile device an acquisition board, developed by our research group was used. The experimental environment makes use of a PS-1500 ICEL source to feed the mobile. Adafruit integrated circuit contained in INA219[1] was used to measure the current. An Arduino UNO[3] was used to read the data from this integrated circuit. After reading, data is sent through a serial port to the computer and there the data is stored by using PCommSerial [15]. We used a smartphone Samsung Galaxy IV with Android 4.3 Jelly Bean

that used 3G data network from the Claro provider. The figure 1 shows the experimental environment.

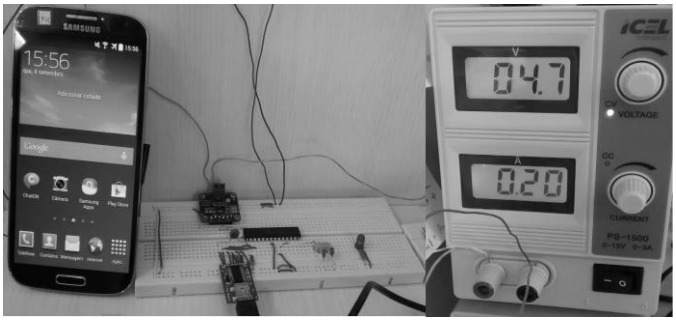

Fig. 1 The experimental environment

V. Case Study

The proposed application is intended to track the world cup games. It consists in showing the scores of games in the application and update the application when there are any changes in scores. The developed application can be seen in Figure 2.

To build the server to simulate games and send the data to the GCM server, a machine from the Informatics Center (CIn) was used within the network of the Federal University of Pernambuco (UFPE).

This server was used to simulate the games, but it can easily be modified to rescue results from any site which provides these data, such as Yahoo, CNN, NBC, BBC, among others.

There are basically four components: the game server, the GCM server, the application that uses the technique of polling and that use the technique of pushing. The first server is responsible for simulating the games and sends them to the GCM server and also provides information about games to the application using the technique of polling.

The experiments consist in the measurement of the energy consumption of the applications. The first application measures use the polling technique and the second one use the pushing technique inside the GCM.

The application that uses polling sends a request to the game server to receive updated data and the server returns the data up-to-date to the application. The application using the pushing technique initiates communication with the GCM server to notify that it is waiting for data to update and then the GCM server waits for data from the game server. The GCM only sends messages when there is data to be updated, not requiring mobile application to request. An active connection is maintained, but there is no transmission of data between the GCM and the mobile application.

To prove the gain from the proposed pushing approach, two android applications were built: one using the pushing (using GCM) technique and another one using the polling technique to update the database from the application.

To feed these applications with the game data, a web system (section 5.3) using Java was built. This system simulates games data and sends it to the GCM server that synchronizes data with mobile application that uses the technique of pushing.

Fig. 2 The application running

A. Methodology

The experiment consists in performing the measurement of each application for an hour. Each experiment was repeated 55 times to obtain a consistency in data and more accurate statistical information.

In Figure 3, it is possible to see the flow of the methodology developed in this work. In the first stage, the applications were built, one application using the pushing technique and another application using the polling technique. In the second step, a measurement environment was built. The INA219 integrated with the computer to receive the data, the energy source plugged for the smartphone and an Arduino UNO receiving data from the INA219. The measurement environment takes and stores 138 samples of voltage and current every second.

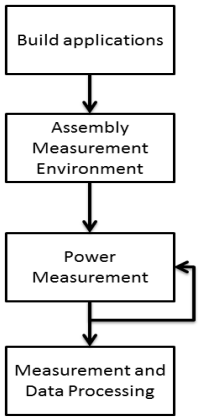

Fig. 3 Research flow

In the third step, the consumption measurement of the smartphone was performed while running the application. Each iteration was repeated 55 times and each experiment had a one hour execution. Everything that could consume battery during the test was disabled, such as GPS, Wi-Fi, data synchronization, scheduled updates, among others. This was necessary to minimize the fluctuation in energy measurement from routine applications of the O.S., and thus do not alter the measurement of the application.

The fourth and final step consisted of a treatment of the collected data, calculation of the energy consumption for each iteration of drawing graphs linking relevant data to this work.

To calculate the energy consumption, the trapezoidal rule [20] was used to calculate the area of the electric power graphic. The instantaneous power is calculated by P (t) = v (t) * i (t), in which i(t) is the current and v(t) is the operating voltage for a certain time t. The voltage data and current are obtained by reading the INA219 circuit by Adafruit. The microcontroller sends this data to a server computer and it stores the data in files. Each experiment has a title with the number of the experiment file.

B. Polling Application

Figure 4 shows the overall flow of the application that uses the technique of polling. After the application startup, the games are immediately loaded from database to be viewed on the device. After that, one thread to make HTTP requests to update the scores. Each new score updates the database and consequently the view on the device.

When the database is empty, the procedure Download Teams and Download Games are executed the first time the application runs. The first one is responsible for downloading the name of teams and the second one to download the scores and locations of games. A thread responsible for request the server to check if there are data to synchronize is performed every minute and is active while the application is running.

An Android service [2] to start the thread was used. Even with the application in background, the service runs normally. You must run the correct wake up procedures in the operating system for the request not to be destroyed by the O.S. This service is responsible for waking up O.S. every minute, making the request, updating the database, and then updating the view of the application.

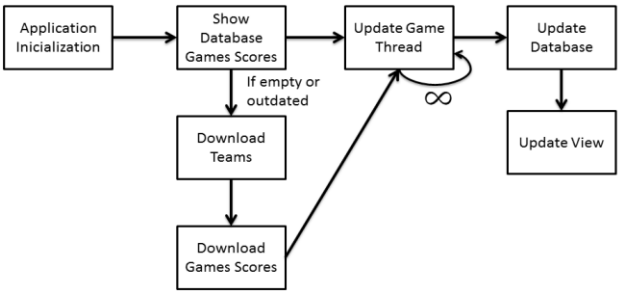

Fig. 4 Application flow using polling

C. Pushing Application

Figure 5 shows the application flow using the pushing technique. After the application startup, the games are loaded from the database to the view of the device, as occurs in the application using polling. After that, there is a negotiation connection with the GCM server. This connection is kept alive by a service (android service) and even with the application closed the service continues to run to be alerted when there is any data to be transmitted from the GCM to the device. When you have some data to be updated, the score of a game, for example, the service receives this data, updates the database and then updates the view of the application.

The pushing technique has another very striking advantage. There is no need to keep activity on foreground to maintain the application. It can be closed and still be updated. The service that communicates with the GCM and updates the data is active from the application start until kill the process or restart the smartphone.

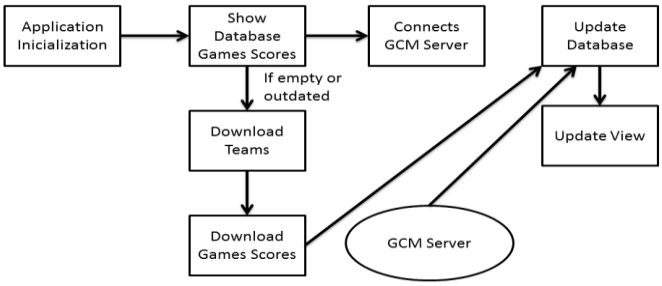

Fig. 5 Application flow using pushing

D. Java Server

The Java server consists of a system responsible for simulating games of the world cup and sends this data to the GCM. This server is also used by applications to update the database at first access. The GCM has a delay between the data arrival and sends this information to synchronize with the mobile application. Even with this short delay, it'll be noticeable to the user when he opens the application for the first time, because nothing will appear. To solve this problem , the first update of the database is made when the application is started sending an HTTP request to the Java server that returns the names of the teams, current scores, local games and links URL pointing to the flag of the team.

To communicate with the GCM server, a prior registration is required on Google platform to generate an ID to be used in the GCM API. The server that sends the data has an ID called ID_Sender and every mobile application also has an ID called Android_Device_ID. With this ID is possible choose which devices must receive the information. This work sends a broadcast to all registered devices.

VI. RESULTS

Figure 6 shows the power dissipated by the application that uses polling in an interval of 5 (five) minutes. Each peak of power indicates a request. Seven clearly distinct peaks can be seen on the graph, which indicates 7 (seven) requests. In an interval of 5 minutes and with the application executing 1 request per minute, there should be only 5 (five) peaks, but there are variations, such as network congestion, which lead to the repetition of the request, imprecise thread count time, Android O.S. that can delay the request or internal policies in Android energy saving.

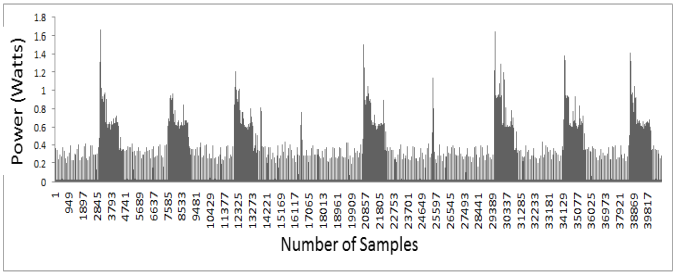

Fig. 6 Consumed power using polling

Figure 7 shows power from the application using pushing in a range of time of approximately 5 (five) minutes. Three distinct peaks can be seen. The first and second peaks are easily visible between samples 6500 to 11000 and 14000 to 19000. These peaks represent data traffic to update the database of the application, i.e. a transmission of information between the GCM server and the mobile application. The third peak can be seen between samples 24000 and 24500, this peak is a keep-alive, a sign sent by the HTTP protocol to keep connection active. The other points on the graph are noises generated by the O.S. or maintenance of the data network.

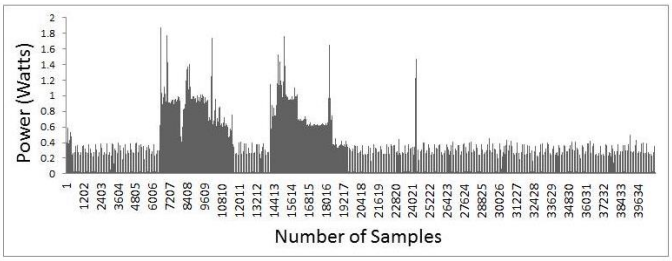

Fig. 7 Consumed power using pushing

Figure 8 shows the figures of the total energy consumption for all experiments. It can be clearly seen that the application using the polling is always higher than the one pushing, proving a substantial gain. The average consumption for the application using polling in 447 J with a standard deviation of 41 J was calculated based on these figures. In the application using pushing, the average consumption was of 156 J with an average standard deviation of 15J.

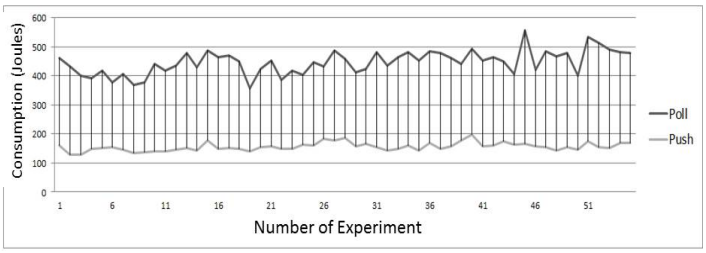

Fig. 8 Energy consumption of polling and pushing approaches

Linearity in energy gain in the pushing application against polling application can be seen in Figure 9. The average gain is of 187%, a significant value when compared to other works consulted. There are differences in gains because 3G network is very fickle, may have traffic congestion and processes running in O.S. background. This interferes in the general consumption of the application and a statistical analysis was needed. The standard deviation of the average gain is of 26 percentage points.

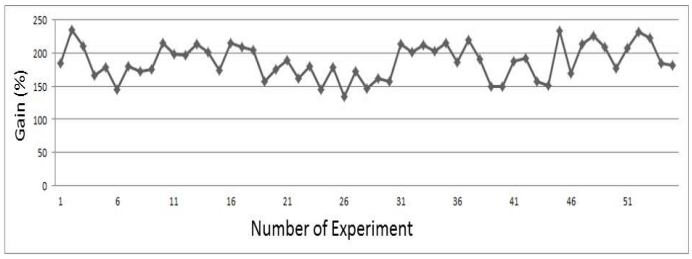

Fig. 9 Gain percentage in pushing approach

After proved the energy efficiency of the pushing approach in the proposed case study, this paper investigated the variables of the polling technique. Energy analysis was also made in requests with different times. In the previous example, the polling was applied every 1 minute interval and it was also applied for intervals of 5, 10, 15, 20 and 30 minutes. The results can be seen in Figure 10. The consumption figures of the polling approach can be seen in Table I. The pushing approach doesn't have variations in time requests.

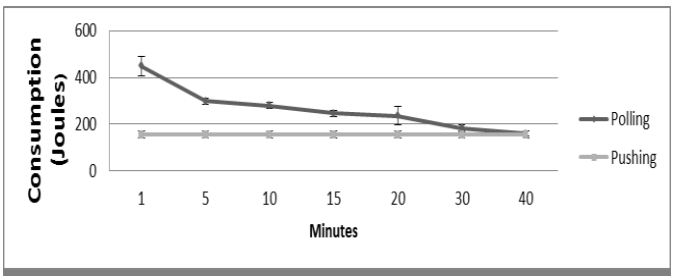

Fig. 10 Consumption in many requests time

With these results we can specify when to use the most appropriate approach. If the application doesn't need to make more than one request in the range of 40 minutes or more, it's advisable to use the polling technique to be more energy

efficient. That is possible due to the fact that the pushing approach has a long-term connection established with the server to maintain the data synchronization awake and it also has a stable energy consumption.

TABLE I
ENERGY CONSUMPTION FOR DIFFERENT TIME REQUESTS

Request Time / Approach	Polling (arithmetic mean)	Standard Deviation
1 minute	447.86 J	41.18 J
5 minutes	298.94 J	13.12 J
10 minutes	280.63 J	14.19 J
15 minutes	245.76 J	13.94 J
20 minutes	236.35 J	37.77 J
30 minutes	183.38 J	15.76 J
40 minutes	159.38 J	11.02 J

Finally, this work proved the effectiveness of the pushing approach for data synchronization compared to polling technique in a case study and shows when it is better to use polling. With this information, it is possible to choose the best approach for data synchronization on mobile devices.

VII. CONCLUSION

Literature was examined to understand how applications synchronize their data with their servers and try to improve energy consumption in this area of knowledge. This work used two techniques from literature, the polling and pushing for data synchronization between applications. An experimental environment based on the Arduino platform and INA219 to get the samples was used. In this paper two applications were built to evaluate the energy consumption by using the two techniques. Various measurements were performed using a real environment and results showed an energy gain of approximately 187% whenever using the pushing technique compared to the polling technique in the analysed case study. A more detailed analysis found that if an application using polling does not make more than one request for data synchronization in the range of 40 minutes or more, it is strongly recommended to use the polling technique for energy consumption saving.

ACKNOWLEDGMENT

To teacher Abel Guilhermino for providing the conditions for the development of this work. The Informatics Center of UFPE for encouraging students to do research and offer subsidies for the development. The authors would like to thank CNPq (Universal/472317/2013-0) and FACEPE (IBPG-0731-1.03/12) for partial financial.

REFERENCES

[1] Adafruit Industries. “INA219 High Side DC Current Sensor Breakout - 26V ±3.2A Max”. http://www.adafruit.com/products/904. Mar 2014.

[2] Android Services. http://www.tutorialspoint.com/android/android_services.htm. Jan 2014.

[3] Arduino. “Arduino Uno Board”. http://arduino.cc/en/Main/arduinoBoardUno. Mar 2014.

[4] Cisco. “Cisco Visual networking Index: Global Mobile Data Traffic Forecast Update,” White Paper, Feb 2014.

[5] S. Deng and H. Balakrishnan. Traffic-aware techniques to reduce 3g/lte wireless energy consumption. In Proc. CoNEXT (2012), pp. 181–192. 2012.

[6] N. Fernando, S. Loke, and W. Rahayu. “Mobile cloud computing: A survey”. Future Generation Computer Systems, 2013, pp. 84–106. 2013.

[7] Gartner. “Gartner Says Worldwide Smartphone Sales Soared in Fourth Quarter of 2011 With 47 Percent Growth”. http://www.gartner.com/newsroom/id/1924314. 2012.

[8] Gartner. “Record Smartphone Shipments Grow the Market 38.8% in the Third Quarter of 2013, Making Way For A Strong Holiday Quarter, According to IDC”. http://www.idc.com/getdoc.jsp?containerId=prUS24418013. Nov. 2013.

[9] Google Android. “Portal Desenvolvedores”. http://developer.android.com/guide/components /services.html. Jan. 2014.

[10] GSM Association. “Fast Dormancy Best Practises”. http://www.gsma.com/newsroom/wp-content/uploads/2013/08/TS18v1-0.pdf. Jan. 2014.

[11] J. Hansen, T. M. Grønli, and G. Ghinea. “Towards cloud to device push messaging on Android: technologies, possibilities and challenges,” International Journal of Communications, Network and System Sciences, Vol. 5, No. 12, pp. 839-849, 2012

[12] J. Huang, F. Qian, Z. M. Mao, S. Sen and O. Spatscheck. “Screen-off traffic characterization and optimization in 3G/4G networks”. Proceeding in: Conference on Internet measurement Conference IMC '12. pp. 357-364. ACM New York, NY, USA, 2012.

[13] IDC. “Despite a Strong 2013, Worldwide Smartphone Growth Expected to Slow to Single Digits by 2017, According to IDC”. http://www.idc.com/getdoc.jsp?containerId=prUS24701614. Feb. 2014.

[14] J. P. Martin-Flatin. “Push vs. pull in Web-based network management”. Proceedings of the Sixth IFIP/IEEE International Symposium on Integrated Network Management, 1999. Distributed Management for the Networked Millennium.

[15] PcomSerial. “Moxa PComm Lite - Serial Comm Development Tool”. http://www.moxa.com/product/download_pcommlite_info.htm. Mar. 2014.

[16] F. Qian, Z. Wang, A. Gerber, Z. M. Mao, S. Sen and O. Spatscheck. Characterizing radio resource allocation for 3G networks. In Proceedings of ACM IMC, 2010.

[17] A. G. Silva-Filho et al. “Energy-Aware Technology-based DVFS Mechanism for the Android Operating System”. Proceedings of the IX Operating Systems Workshop, Brazilian Symposium on Computational System Engineering, Natal, Brazil, 2012.

[18] Statista. “The Statistics Portal”. http://www.statista.com/statistics/278305/daily-activations-of-android-devices. Jan. 2014.

[19] S. Taleb, M. Dia, J. Farhat, Z. Dawy, and H. Hajj. “On the Design of Energy-Aware 3G/WiFi Heterogeneous Networks Under Realistic Conditions”. Published in: 27 th International Conference on Advanced Information Networking and Applications Workshops (WAINA), 2013.

[20] E. Talvila and M. Wiersma. “Simple Derivation of Basic Quadrature Formulas” http://arxiv.org/pdf/1202.0249v1.pdf. Feb. 2012.

[21] F. Xu, Y. Liu, T. Moscibroda, R. Chandra, L. Jin, Y. Zhang and Q. Li “Optimizing Background Email Sync on Smartphones”. Proceeding of the 11th annual international conference on Mobile systems, applications, and services pp 55-68 ACM New York, NY, USA, 2013.

[22] P. C. Dinh and S. Boonkrong. The Comparison Of Impacts To Android Phone Battery Between Polling Data and Pushing Data. International Conference on Computer Networks and Information Technology – ICCNIT. Bangkok, Thailand, 2013.