Energy Consumption in Mobile Devices Considering Communication Protocols

[1]Verônica Conceição Oliveira da Silva, [2]Danilo Mendonça Oliveira,
[3]Jean Carlos Teixeira de Araujo, [4]Paulo Romero Martins Maciel
[1.]*Informatics Center, Federal University of Pernambuco, Recife, Brazil, vcos@cin.ufpe.br*
[2.]*Informatics Center, Federal University of Pernambuco, Recife, Brazil, dmo4@cin.ufpe.br*
[3,4.]*Informatics Center, Federal University of Pernambuco, Recife, Brazil, jcta@cin.ufpe.br, prmm@cin.ufpe.br*

Abstract

Smartphones are popular and essential today, and they used in multiple everyday activities. Internet is an essential resource to mobile applications that, usually, involve data synchronization with remote servers and other clients, by using wireless communication. The data traffic reflects in energy resources consumption of mobile device. In instant message applications, several protocols may be used to perform data synchronization between mobile client and server. This paper investigates the impact in energy consumption produced by server push technologies on mobile devices. The most common protocols used in these types of applications are short polling, Comet, WebSocket and XMPP. A workload generator was implemented to test each protocol. The availability of each scenario has been estimated by using hierarchical heterogeneous models. We evaluated the energy consumption overhead of those protocols in both Wi-Fi and 3G connections. The results indicate that XMPP is the most efficient protocol in terms of energy consumption.

Keywords: *Energy Consumption, Availability Models, Mobile Networks*

1. Introduction

Nowadays, the information is no longer just stored on a local device; it is possible to access data and information anytime and anywhere. In addition, there is an ever-increasing need for communication between people [19].

The rapid smartphone and tablet markets growth is modifying the Internet scenario. In 2012, Hollister [12] stated that mobile devices generated more than 10% of the Internet traffic. The mobile applications market is growing, with more and more applications being developed to assist and facilitate human activities [13]. People want and expect more resources for their devices, and an increasing need for instantaneous communication creates opportunities in applications development.

Mobile phones have evolved-to work-faster, capture high-quality images, and numerous other features. As a consequence, the apps market has developed at an even greater pace with the spreading of mobile Internet access, resulting in significant data traffic. However, mobile devices have limited energy resource. Optimizing applications is very important to save battery energy since the quality requirements for these devices are continuously growing [4].

Greater the number of applications and resources a mobile device has, more energy will be to perform these applications and maintaining these resources [10]. Develop applications that preserve device's energy is a hard task, since performance is a critical aspect in some areas, such as games, computer vision, mHealth, etc. In the domain of instant message applications, it is required an always on a wireless connection to sending and receiving messages. This load on device's wireless interface increases the energy consumption significantly.

Availability models assist to estimate which critical system points must receive more attention. The model Reliability Block Diagram (RDB) used to availability each mobile system component, including hardware and software, which allows understand criticality each component and work to improve it. To represent discharge battery, used model Continuous-Time Markov Chain (CTMC). The model can grow with new battery resources, like more autonomy. This model assists to represent what happened in each discharge phase with a variety of data network connections, some papers used CTMC like a model resource to represent discharge energy stages.

Many protocols were developed to permit the message exchanging of clients across the Internet. Choosing the correct protocol to use in mobile application could have a significant impact in energy consumption and provide better user experience. In order to understand energy consumption impact caused by each protocol, it is important to evaluate application's performance when using those protocols under the same conditions.

This paper evaluates the energy consumption in mobile devices when using server push technologies. This study considered four communication protocols: short polling, Comet (long-polling), WebSocket and XMPP, combined with two wireless data transfer modes: Wi-Fi and 3G. We performed measurements on a mobile application working with each combination protocol and data transfer mode. To perform those measurements, we developed a test-bed composed by a workload generator that operates with those four protocols. An Arduino board was used to measure the energy consumption during the experiments. Analytical models were proposed to availability mobile device environment.

This paper organized as follows. Section 2 discusses some related works about mobile energy consumption. Section 3 provides an overview about server push technologies and common protocols. Section 4 general explains about methodology adopted, and Section 5 explains details about architecture experiment. Section 6 shows analytical models developed to represent energy consumption in this study. Section 7 shows experiment results obtained through model analysis. Finally, Section 8 concludes this study and proposes future works.

2. Related works

Because of the limitations imposed on mobile device by their batteries, studies are needed to assess which strategies should be applied to improve the energy efficiency of mobile devices. Some works in literature adopt a software-based strategy to evaluate energy consumption in mobile applications [4]. Our study does not follow this approach and employs a hardware component to obtain more accurate results.

According to [18], in comparison with other mobile device components. HSDPA (High-Speed Downlink Packet Access) and wireless LAN represent the first and second largest sources of energy consumption of a mobile device, respectively. This work found that the wireless interfaces provoke more energy consumption than other components like the device's screen and the CPU.

Kalic et al. [15] performed some tests with data transfer in mobile devices and stated that Bluetooth was the most energy efficient communication technology, in comparison with Wi-Fi and 3G. Wi-Fi and 3G connections showed the same battery life usage. However, Wi-Fi can transfer more than twice as many files as 3G, whereas Bluetooth is not suitable for large file transfers and becomes even less energy-efficient with long-term use.

Some works related to server push technologies include investigations of network traffic and server resource utilization. Schneider *et al.* [22] did a statistical analysis of the data traffic generated. It by Web 2.0 applications and found that the automatic refresh features in AJAX-enabled applications were responsible for generating short bursts of large amounts of data.

Bozdag *et al.* [5] evaluated the existing trade-offs between client polling and server push strategies. The authors concluded that the server push approach was essential to maintaining data coherence between the server and the client. However, the workload for the CPU was seven times greater compared to that in client pull. Neither work considered the use of these technologies in mobile applications nor did they evaluate the usage of XMPP and WebSocket protocols as possible alternative communication channels.

In [1], short and long-polling techniques were assessed in terms of scalability and efficiency. It to increase the system scalability, the strategy of turning off the server push mechanism in periods of low message reception probability it suggests. Message receipt times are used to predict when the mechanism must be enables. As opposed to the current work, the authors relinquished the requirement to deliver messages in real time in favor of a probabilistic approach that activates the connection at pre-determined times. Therefore, this strategy would not be suitable for life-critical systems, such as telemedicine and mHealth applications.

3. Background

3.1. Reliability block diagram

RBD is a logical flow input and outputs required of the system and are an important modeling resource used to determine the device's reliability.

System reliability block diagrams and our models start with necessity decompose each component of the system. In the diagram, each block represents one component that together comprise the system. Usually, we use RBD model to demonstrate how determine the cost and critical level of each component to provisioning plan to repair, maintenance or provisioning spare components. Through statistical models can be seen all system components, and observe most vulnerable components and further studies on the impact this component over system availability [6].

3.2. Continuous-time Markov chain

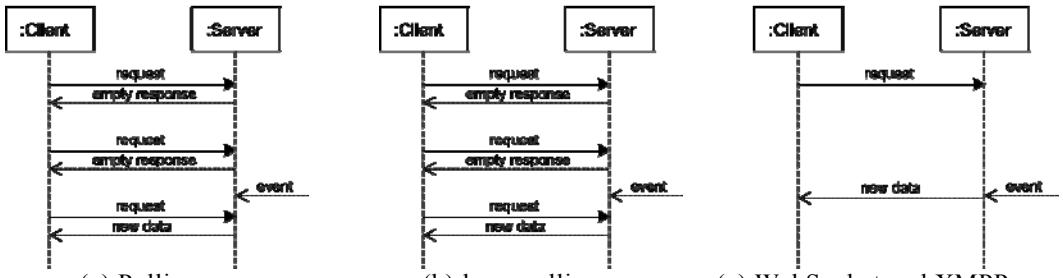
Stochastic processes were discrete in both time and space, and the Markov taken over this property: the process future behavior only depends upon the current state and not any of the rest of the past. Continuous-Time Markov Chain is adopted to generalize models with time continuous and consider the state space to be either unit or countable infinite. In this model, discrete time Markov, transitions can happen at any time [2].

A Markov chain was used to model the environment context because, in Markov processes; random variables do not depend on the history, as was characteristic of the values extracted in the real experiment. In random variables, the real number is assigned to each random experiment environment. CTMC allows expressing distinct scenarios and situations, a flexible tool modeling to analyze the availability the study object.

3.3. Server push technologies

The Web initially designs as a traditional client-server architecture, in which the client begins communication to fetch data. However, when Web 2.0 emerged, the Internet ceased to be a static document repository and became a highly dynamic environment. Nowadays, Web systems are increasingly focusing on solutions wherein client's interact and exchange messages in real-time. This new requirement to supply real-time data has led developers to adapt an existing model. The following subsections describe the main techniques used to achieve that purpose.

Figure 1 illustrates the behavior of each synchronization strategy in a UML sequence diagram. In Fig. 1-a, which is a diagram of short polling, there are two requests made that receive empty responses because there is no update on the queried data. After receiving the "event" asynchronous message in the server, a data update occurs, and the updated data receive as a response to the next request. In Fig. 1-b which depicts the long-polling technique, the request-response remains pending until the server has new data to deliver. After it is receiving updated data, the client opens another connection, thus starting a new cycle. Figure 1-c illustrates the strategy when there is a full-duplex connection between the client and the server (WebSocket and XMPP). Here the server has the freedom to send data to the client while the connection is open.



(a) Polling (b) long-polling (c) WebSocket and XMPP

Figure 1. Sequence diagrams for the synchronization strategies

3.3.1. Polling

This strategy, also described as client pull or short polling, consists of periodically querying the server for updated data. If the polling interval can be adjusted according to the server data update rate, an updating mechanism approaching real time is possible.

There are several disadvantages to the utilization of this technique. Each polling operation creates an overhead caused by the HTTP header in every request, which implies high consumption of server resources in multi-user environments where the request volume may be great. Additionally, if the data update rate is slower than the polling interval then unnecessary requests will be made.

3.3.2. Long-Polling

In Long-Pooling to circumvent the empty-response request problem, an extension to the original polling mechanism proposed. In the long-polling technique, the client requests as before, but the server does not send an empty response when there are no data to be was sent. Instead, it keeps the connection open until a message addressed to that particular client arrives. When that happens, the server writes the data and closes the connection. When receive the message, the client sends another request, thus continuing to be updated if and when messages arrive. When utilized along with AJAX, this technique is known as *Comet* [20].

The main improvements achieved with this strategy are a reduction in message latency and the elimination of empty response messages. In terms of thread creation, it is important to remember that each request creates a new thread. In long polling, N users imply exactly N threads in execution at any one time. In short polling, the number of simultaneously running threads created by a typical 1 second polling strategy may be many times greater than N. In a multi-user environment, short polling could even cause system unavailability. If the server resources, become sufficiently depleted due to the prohibitive volume of running threads.

3.3.3. WebSocket

Short and long-polling strategies possess certain characteristics that are disadvantageous in terms of scalability for the applications that use them [8]. For instance, each client requires two TCP connections: one for data transmission and another for message notifications. There is also the overhead caused by the HTTP header contained in each message.

It to counter such disadvantages, the WebSocket protocol was created to provide a full-duplex channel for browser applications through a single TCP connection. The WebSocket protocol, together with the WebSocket API [11] consists of a scalable and robust solution for real-time Web applications, which overcomes the limitations of previous alternatives. As opposed to long-polling strategies, a new request does not need to be sent by the client after receiving data, nor are HTTP headers added to messages.

3.3.4. XMPP

Extensible Messaging and Presence Protocol (XMPP) is an XML-based protocol for instant messaging that supports contact lists and online status information. It creates under the name Jabber in 1999 [14], and it accepted as an IETF standard in 2004 [21]. Since its approval by the IETF, XMPP has been gaining support from large companies such as IBM, Google, and Apple, and has become the main open standard for message exchange in real time. It has also received several extensions [17] that allow the utilization of the protocol in a wide range of applications.

4. Methodology

This work investigates energy consumption in the real scenario using variety protocols combine two distinct data transfer. The Figure 2 shows environment creates to analyze energy consumption.

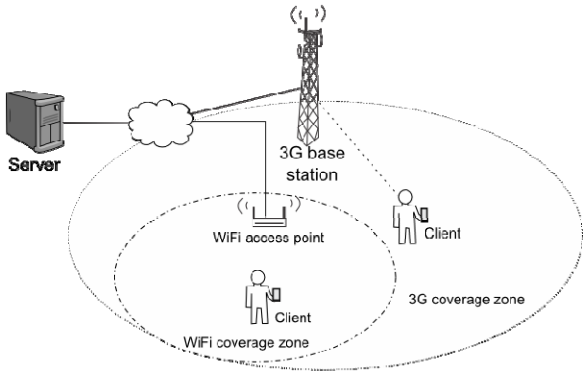


Figure 2. Scenario architecture

We built a workload generator that runs on a server, and can send instant messages to a mobile application using four different protocols: short polling, Comet, WebSocket and XMPP. In our experiments, we considered the sending of messages in both data transfer modes: Wi-Fi and 3G. The obtained results were use for system modeling and comparisons between different scenarios.

The environment was modeled to analyze each part of the study. It was used Reliability Block Diagram (RBD) model to represent mobile system and determined critical level each component. So it is possible understand the availability of the mobile system used in this environment and increase the study in critical aspects that impact in usability in mobile resources. To battery discharge was used Continuous-Time Markov Chain (CTMC) to represent each phase, and probability to choose one kind data transfers. Some other papers chose to use Petri Nets [16]. However, we decided to use CTMC to represent battery discharge model, in another point of view.

Those models were parameterized by the experimental results. To compose analytical models and evaluate this study, we used two software packages: SHARPE tool (version 1.3.1) and Mercury tool (version 4.1.2).

5. The environment

The environment builds over workload generator and mobile client application to receive messages. It was performed some monitoring experiments on a real system. We used two varying parameters on the experiments: the server push protocol and the data transfer mode. Each combination of values for those parameters defined an experiment scenario. The list of all considered scenarios is listed on Table 1.

Table 1. Scenarios

Scenario	*Synchronization mechanism*	*Communication*
1	Short polling	Wi-Fi
2	Comet (long-polling)	Wi-Fi
3	WebSocket	Wi-Fi
4	XMPP	Wi-Fi
5	Short polling	3G
6	Comet (long-polling)	3G
7	WebSocket	3G
8	XMPP	3G

5.1. Workload generator

In order to evaluate the protocols in terms of energy consumption, it was developed an asynchronous message exchange application. That uses an architecture based on the publish-subscribe pattern, in which the server provides channels for clients to connect and receive messages. The design includes four different types of communication channels corresponding to the strategies to be evaluating; short polling, Comet (long-polling), WebSocket and XMPP.

Figure 3 illustrates the application architecture. The message exchange system is base on the Java Message Service (JMS) [9] standard and runs on a GlassFish server. Messages published through a call

to the Web service implemented by the *Web Service Notifications* module. The *Polling Server*, *Comet Server*, *WebSocket Server* and *XMPP Server* modules publish the channels that the clients can connect to and receive messages. Each module has a corresponding client-side module: *Polling Client*, *Comet Client*, *WebSocket Client* and *XMPP Client*. The modules are implemented as background services, and deliver messages to the application.

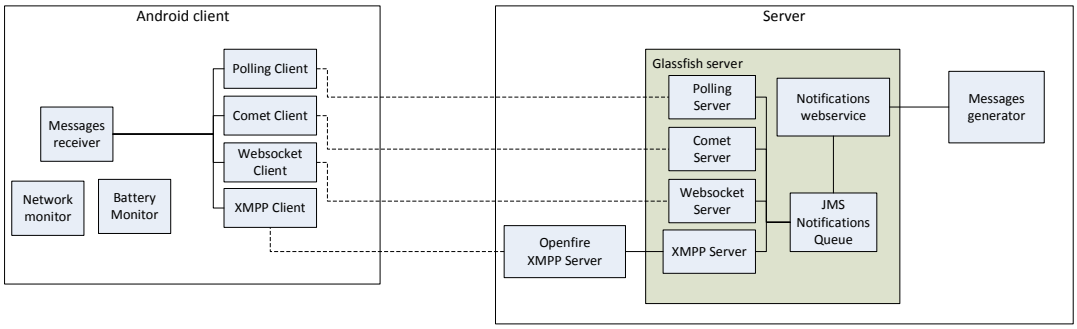


Figure 3. Application architecture

The XMPP channel introduces an additional element: the *XMPP Openfire server*. The client connects to this server to receive messages through XMPP. The *XMPP Server* module then connects to the Openfire server and forwards the received JMS messages to the corresponding user.

The *Message Generator* module is a system workload generator and is responsible to sending messages to the user continuously during the experiments. This module allows configuring the sending rate (messages per second), the payload size, and the destination user. The time between messages distributes exponentially.

Table 2 presents the parameters used in the experiments. The *Synchronization mechanism* and the *Data transfer mode* are the two varying parameters described earlier. The other parameters are fix in these experiments.

Table 2. Parameters

Parameter	***Value***
Synchronization mechanism	Short polling, Comet, WebSocket, XMPP
Data transfer mode	Wi-Fi, 3G
Message size	100 bytes
Average time between messages	4 seconds (exponential)
Polling interval	1 second
Sample interval energy	0.5 second

5.2. Energy measurement

In order to perform energy consumption measurement on the mobile device, it was used an Arduino UNO R3 Board [3]. This hardware allows us to monitor the voltage used at application execution time. The adopted mobile device is the Samsung Galaxy Mini, model GT-S5570B. It uses Android operating system version 2.3.6, Gingerbread compiler, baseband version S5570BVJKPB, 7.2Mbit/s HSDPA throughput and 900/ 2100Mhz 3G band and 850/900/1800/1900Mhz GSM band operating frequencies and battery capacity of 4.4Wh.

Figure 4 depicts the measurement test-bed used in experiments. The mobile device was connected to Arduino board with the aid of a protoboard and resistor. For the collection of the instantaneous power, we used an application read analog voltage. For each experiment, the mobile application was executed with one protocol combined data transfer mode and measured the instantaneous current consumption on a mobile device. A DC power supply was used to avoid fluctuations in the supply voltage to the mobile device, and therefore allowing us to obtain more accurate measurements. The data collection occurred at 0.5-second intervals.

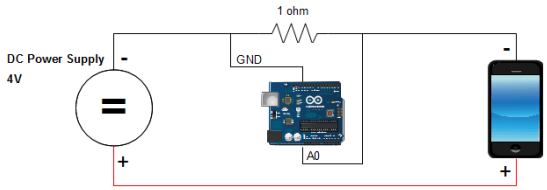


Figure 4. Measurement test-bed

It to understand energy consumption over mobile device it is necessary to understand the Equation 1. This relationship defines the energy consumption rate of an electric circuit, where *V* is the electric potential (voltage) in volts, and *I* is the electric current in amperes. In order to calculate the electrical energy consumption *E*, we used the Equation 2 that relates the power *P* obtained from the previous equation, with a time interval *t*. The obtained value represents the electrical energy consumption in the watt-hour unit within a specified time interval.

$$P(t) = V(t) \times I(t) \tag{1}$$

$$E = \int_0^t P(t) \times dt \tag{2}$$

Through numerical integration, it is possible to measure the overall energy consumption of each scenario in our experiments. By obtaining samples of the power *P* at intervals of 0.5 seconds collected by the Arduino board, we apply the Equation 3 on those samples to calculate the total energy consumed during a specified interval. With this method, we can comprehend the power consumption during the experiment and predict application autonomy when combine Wi-Fi or 3G networks with each server push protocol. For more accuracy, we performed each experiment several times to obtain differences in energy consumption between the same scenarios.

$$F(x) = \int_0^{\infty} f(x).dx \simeq \sum_{k=0}^{n} f(x_k).t \tag{3}$$

This paper does not consider linearity on battery consumption, nor does it take into account consumption of other applications in the mobile device. It was turn off many applications of mobile device. However, some system applications such as operational system services could not be disabled. Therefore, in the experiments, the mobile device operated with minimum possible applications in order to avoid external influences of data traffic and CPU usage caused by those applications.

6. Proposed models

This section describes the formal models adopted in our work. We used a Continuous-Time Markov Chain (CTMC) model to represent the battery discharge behavior of mobile client. Reliability Block Diagrams were used to evaluate the system availability.

6.1. Battery discharge model

Modeling techniques were applied to evaluate mobile device energy consumption in this context. We opted battery discharge modeling by Continuous-Time Markov Chain (CTMC). Figure 5 estimates the availability of each protocol in mobile devices and their possible energy consumption states when choose the likelihood of one data connection.

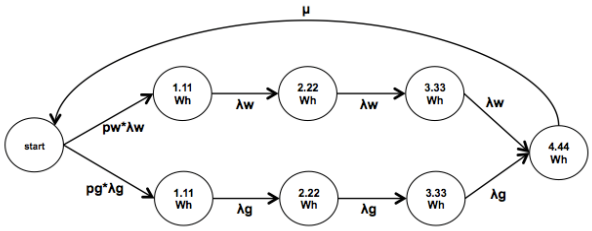


Figure 5. CTMC for the battery discharge

The starting stage considers when mobile device was turn on, i.e.; the discharged battery begins. In this stage, the battery is fully charged. Thus, the user starts to use a mobile device and selects an instant message application. Two data transfer mode are available: Wi-Fi and 3G. When use mobile device data transfer begins to discharge the battery, we have two different rates: λw, the discharge rate when using Wi-Fi connection and λg, the discharge rate when using 3G connection. The first stage must also consider the probability choose of data connection between Wi-Fi (*pw*) and 3G (*pg*).

When model evolution reached state 4.44Wh, the battery was altogether discharged. It to return into the initial condition; time is needed to recharge the battery; this is represented by μ rate. During application's executing, discharge battery occurs, in some time battery lost 1.11Wh energy, after some time 2.22Wh, 3.33Wh and totaled 4.44, all accumulated energy in the battery. In this model, the discharged battery was represented in four stages. With this would be possible to use the same CTMC model to battery with less accumulate energy, or just add more stages to more accumulate battery energy. The value 4.44Wh was chosen because represents the total battery capacity for a mobile device tested.

To determine the availability of the mobile device, it by using the CTMC following equation:

$$1 - \frac{(pg\lambda g + pw\lambda w)\,\lambda g\,\lambda w\,\lambda wg}{(pg\lambda g + pw\lambda w)\,\lambda g\,\lambda w\,\lambda wg + \left((3\,pg\lambda g + \lambda g)\,\lambda w\,\lambda wg + pw\lambda w\,\lambda g\,(\lambda w + 2\,\lambda wg)\right)\mu} \quad (4)$$

6.2. Reliability block diagram

We adopted the reliability block diagram (RBD) to modeling mobile device. Figure 6 shows the RBD model used to estimate mobile phone reliability. It consists of the following components: mobile hardware (*mobile_hw*), mobile operation system (*mobile_os*), mobile application (*mobile_app*) and mobile battery (*battery*).

Figure 6. RDB mobile device

In RBD shows in Figure 6 some essentials components were considered to analyze the availability of the mobile device. These components represent more common components of a mobile device. The availability from the customer viewer point, it is determined through Equation 5:

$$A_{mobile} = A_{mobile\_hw} \times A_{mobile\_os} \times A_{mobile\_app} \times A_{battery} \quad (5)$$

7. Case studies

The case studies of our work consisted of the following steps. First, we performed experiments in order to obtain the consumption rates of CTMC model associated with each protocol using different wireless interfaces. Then, we compared the different scenarios and determined the most efficient

protocol in terms of energy consumption. Lastly, we chose the discharge rate of this protocol to be adopted by the RBD model, and then evaluating availability and reliability of the mobile system.

We adopted an execution time of one hour to each experiment. In this time, we collected a total of 7,200 samples of instant power consumption. With those samples, we calculated the energy consumed in the experiment by applying Equation 3. The workloads generated while the experiments and the monitoring mechanism adopted described in Section 4.

Figure 7 shows the energy consumption of mobile clients using Wi-Fi wireless connection. We found that the *Comet client* consumes more energy than the other technologies, even more than the *short polling* protocol that was supposed to be the most inefficient.

The *WebSocket client, XMPP client*, and *Polling client* showed distinct behaviors. The *XMPP client* was more energy efficient than others with Wi-Fi data transfer mode.

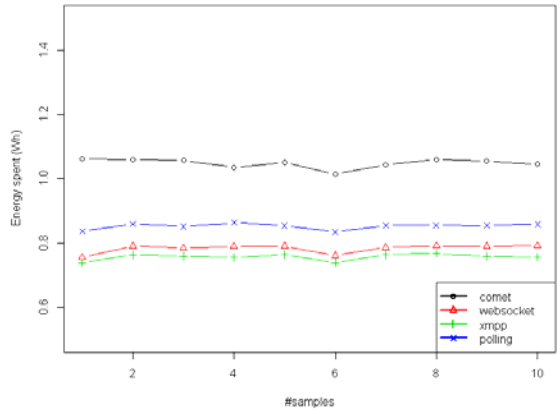


Figure 7. Energy consumption of push clients that use Wi-Fi connection

Figure 8 shows the energy consumption of each protocol operating with a 3G wireless connection. In this scenario, the mobile device was fixed to conserve the same 3G connection properties. Without variables in others technologies data transfers like GPRS, EDGE or 4G, the mobile device was associated to the same base station avoiding exchange in cells during the experiment. Any traffic with voice service was blocked to avoid interference the results. The results showed that the *Polling client* consumes more energy than others, and the *WebSocket client* was the most efficient. The *Comet* and *XMPP* clients had intermediate energy consumption.

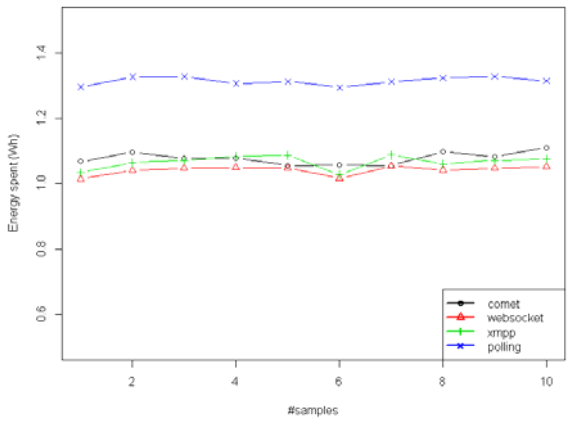


Figure 8. Energy consumption of push clients that use 3G connection

Considering that a mobile user will also be subject to both data transfer type. It is interesting a study to review which of the protocol is the most efficient to both wireless interfaces. We used CTMC model

to check all protocols, considering both wireless interfaces. Looking for greater availability, we consider same probability to both wireless interfaces.

Table 3 describes the rates used in the CTMC model. Each rating value considered as the average of ten experiments lasting one hour. The fully charged battery contains 4.44Wh of accumulated energy. If we find how much energy (Wh) the mobile device spends in one hour, we can estimate the battery discharge rate.

Table 3. *Rate of measurements obtained in the experiments*

Scenario	*Average*	*Rate (λ)*
Short polling Wi-Fi	0.852187222	0.04798351
Short polling 3G	1.314640556	0.07402255
Comet (long-polling) Wi-Fi	1.048572778	0.05904126
Comet (long-polling) 3G	1.078425000	0.06072212
WebSocket Wi-Fi	0.782936111	0.04408424
WebSocket 3G	1.041268333	0.05862997
XMPP Wi-Fi	0.756508333	0.04259619
XMPP 3G	1.066531111	0.06005242
Recharge	2:53	0.34682081

Table 4 shows the availability results of the mobile device obtained by the CTMC model with different protocols. The *XMPP* protocol has better availability results, followed by the *WebSocket*, *Comet* and *Short polling* protocols.

Table 4. *Availability results*

Scenario	*Availability (%)*	*Ranking*
XMPP	0.964323621	1
WebSocket	0.964301635	2
Comet(long-polling)	0.962297821	3
Short polling	0.957879159	4

To parameterized RBD model, it is necessary to determine the mean time to failure (MTTF) and mean time to repair (MTTR) of each component. Table 5 shows the parameters adopted for mobile system in our analysis. The MTTF for the *mobile_hw* and *mobile_app* components obtained from data available in [16]. The MTTF for *mobile_os* component was adopted from [7], and the MTTF of *battery* was obtains through the CTMC model.

Table 5. Mobile system parameters

Components	*MTTF(hours)*	*MTTR(hours)*
mobile_hw	224,719.1	12
mobile_os	1,440.922	0.03333
mobile_app	336.7	0.01666
mobile_battery	4.613	2.883333

.

8. Conclusion

This paper investigated the impact of the four most common instant messaging protocols on the energy consumption, availability and reliability of mobile applications. It developed a workload generator that implements all four protocols to simulate a real message traffic environment and evaluate their impact on energy consumption. Polling protocol showed a high impact in the energy consumption. It was the worst protocol performed in 3G data transfer, and the second worst performed in Wi-Fi connection. Long-polling protocol (Comet) is an extension to the original polling mechanism. It expected that it presented better results than polling, like happened in 3G scenario. However, it showed the worst results in Wi-Fi connection, i.e., it expends more energy than pooling, the opposite expected. WebSocket proved to be the most efficient protocol when used in a 3G connection. However, XMPP protocol has consumed less energy than WebSocket in a Wi-Fi connection. To identify, the

most efficient protocol to mobile devices. Should be considered transitions between data connection networks, because the mobile device does not stop in the same place, and mobile applications need the best energy performance between both data connection. A continuous-time Markov chain model was used to evaluate the availability in the scenarios considering a transition between Wi-Fi and 3G connections, both with the same weight, 50% of change. Through this model, it was possible conclude that the XMPP protocol is the best in energy saving when data connections are mixed, i.e., it expends less energy than other protocols studied in this paper. Energy saving is an important topic to mobile technologies, but other aspects should be considering in a mobile system reliability. RBD model represents the reliability of the mobile system considering other essential parts to mobile device. Each component should be an appropriate study to increase its performance and to save it failures.

This study focused in energy consumption because the battery is a critical factor to mobile device and your applications. More studies are important to optimize energy consumption. In future work, we intend to include more protocols and investigate their behavior by exploring another parameter. A full-factorial analysis will be used to complement such study.

9. References

[1] Agarwal, Sachin. “Toward a push-scalable global internet”, Computer Communications Workshops (INFOCOM WKSHPS), 2011 IEEE Conference on, pp. 786–791, 2011.

[2] Anderson, David F. and Higham, Desmond J., "Multilevel Monte Carlo for continuous time Markov chains, with applications in biochemical kinetics", SIAM: Multiscale Modeling and Simulation, vol. 10, pp. 146-179, 2012.

[3] Arduino, “About arduino”, Dec. 2013. [Online]. Available: http://www.arduino.cc/

[4] Balasubramanian, Niranjan; Balasubramanian, Aruna and Venkataramani, Arun, “Energy consumption in mobile phones: A measurement study and implications for network applications", IMC ’09 Proceedings of the 9th ACM SIGCOMM conference on Internet measurement conference, pp. 280–293, 2009

[5] Bozdag, Engin, Mesbah, Ali and van Deursen, Arie, “A comparison of push and pull techniques for Ajax”, in Web Site Evolution, 2007 (WSE 2007). 9th IEEE International Workshop on, pp. 15–22, 2007.

[6] Raheja, Dev and Gullo, Louis J., ”Design for Reliability”, John Wiley & Sons Inc., USA, 2012.

[7] Kim, Dong Seong; Machida, Fumio and Trivedi, Kishor S., “Availability modeling and analysis of a virtualized system", Dependable Computing, 2009. PRDC ’09. 15th IEEE Pacific Rim International Symposium on, pp. 365–371, 2009.

[8] Fette, I. and Melnikov, A., “Rfc 6455: The websocket protocol”, Status: Internet Draft, Tech. Rep., 2011, Feb. 2012 [Online]. Available: http://tools.ietf.org/html/draft-ietf-hybi-thewebsocketprotocol-17

[9] Hapner, M. and Burridge, R. and Sharma, R. and Fialli, J. and Stout, K., “Java message service", Sun Microsystems Inc., Palo Alto, CA, 2002.

[10] Harrison, Rachel; Flood, Derek and Duce, David, “Usability of mobile applications: literature review and rationale for a new usability model", Interaction Science 2013, Springer Open, May 2013. [Online]. Available: http://www.journalofinteractionscience.com/content/1/1/1

[11] Hickson, I., “The websocket API”, W3C Working Draft WD-websockets-20110929, Sep. 2011. [Online]. Availability: http://www.w3.org/TR/2011/WD-websockets-20110929/

[12] Hollister, Sean, “Mobile devices account for nearly 10 percent of internet traffic, according to statcounter.”, May 2012. [Online]. Available: http://www.theverge.com/2012/5/11/3012957/mobile-devices-account-for-nearly-10-percent-of-internet-traffic

[13] Hughes, Neil, “how c”, Jan. 2014. [Online]. Available: http://appleinsider.com/articles/14/01/13/app-use-surged-115-in-2013-messaging-social-apps-saw-most-growth

[14] Jabber Inc., “Jabber technology”, Jun. 2011. [Online]. Available: http://www.cisco.com/web/about/ac49/ac0/ac1/ac258/JabberInc.html

[15] Kalic, Goran; Bojic, Iva and Kusek, Mario, “Energy consumption in android phones when using wireless communication technologies”, MIPRO, 2012 Proceedings of the 35th International Convention, pp. 754–759, 2012.

[16] Oliveira, Danilo Mendonca; Araujo, Jean Carlos Teixeira; Matos, Rubens and Maciel, Paulo Romero Martins, "Availability and energy consumption analysis of mobile cloud environments", Systems, Man, and Cybernetics (SMC), 2013 IEEE International Conference on, pp. 4086– 4091, 2013.
[17] Saint-Andre, P., "Xmpp extensions", Feb. 2010. [Online]. Available: http://xmpp.org/xmpp-protocols/xmpp-extensions/
[18] Perrucci, Gian Paolo; Fitzek, Frank H.P. and Widmer, Joerg, "Survey on energy consumption entities on the smartphone platform," Vehicular Technology Conference (VTC Spring), IEEE 73rd, pp. 1–6, 2011.
[19] Perry, Mark; O'Hara, Kenton; Sellen, Abigail; Brown, Barry and Harper, Richard, "Dealing with mobility: Understanding access anytime, anywhere.", ACM Transactions on Computer-Human Interaction (TOCHI), vol. 8, issue 4, pp. 323–347, 2001.
[20] Russell, Alex, "Comet: Low latency data for browsers", The Dojo Toolkit, 2006. May. 2013 [Online]. Availability: http://infrequently.org/wp-content/LowLatencyData.pdf
[21] Saint-Andre, P., "Extensible messaging and presence protocol (xmpp): Core", Apr. 2004. [Online]. Available: http://tools.ietf.org/html/rfc3920
[22] Schneider, Fabian; and Agarwal, Sachin; and Alpcan, Tansu and Feldmann, Anja, "The new web: Characterizing Ajax traffic", Passive and Active Network Measurement, Springer, USA, pp. 31–40, 2008.