Towards enhancing mobile user experience with internet and perception metrics

Alfredo Cádiz, Gabriel Hourton and Gabriel Del Canto
NIC Chile Research Labs
Avenida Almirante Blanco Encalada 1975, Santiago de Chile.
{*alfredo, gabriel, gabo*}*@niclabs.cl*

***Abstract*—In the recent years we have observed an impressive growth in the sale of smart phones and mobile internet plans related to them. This new scenario offers new opportunities for software developers which are attracted to target this market. Currently software development is evolving from classical desktop and servers schemas to a mobile world where software systems run in ever changing environments which can affect the user experience. We propose ways to analyse how mobile internet quality can affect the user perception of software quality. We also present an architecture to enable software to be aware of the quality of its communication channels and implement adaptable algorithms to improve user experience.**

***Keywords*-Internet; Quality of service; Computer network reliability; Software quality**

I. Introduction

There is no doubt that mobile devices, such as tablets and smart phones, are taking over the preferences of domestic and corporate users. In 2011 smart phones sales surpassed desktop PC sales [1] and hence the need for mobile internet connectivity [2], [3]. This explosive growth was triggered by the advent of modern mobile operative systems such as Apple iOS and Google Android which made smart phones more attractive and accessible to customers. In October 2012 both Apple and Google claimed they have about 700.000 active applications available in their stores[1]. These facts show how software developers are now more attracted to target mobile devices. In fact, mobile software has the possibility to take advantage of permanent internet connection and additional information such as location, device orientation and user profiling to offer a better user experience, or Quality of Experience (QoE). We understand Quality of Experience as *a person's perceptions and responses that result from the use or anticipated use of a product, system or service* [4].

Internet connectivity is a key element to deliver responsive mobile software, hence good QoE. When mobile internet connection is faulty internet-dependent software becomes faulty, lowering the perception of quality, specially in applications in which online data access is their main feature [5]. Even though mobile operative systems can tell applications when they are connected or not, they lack of ways to warn applications about overloaded or faulty networks. And even when network status seems to be fine, the perception of the user about the running applications depends of several factors.

Software quality perception and Quality of Experience has been already studied [6], [7], but normally they tackle the perception of quality of the complete application, including factors controlled by the programmer. However, few works [8], [9], [10] address the perceived quality of aspects beyond the application such as the internet connectivity status. In this case they study the phenomenon in desktop/portable computers, while we are interested in analysing mobile internet QoS and QoE in smart phones.

Regarding mobile network status, we have stated the need of specialised techniques to measure mobile internet services [11]. This is because the algorithms used for the standard metrics rely on actively testing the connection, consuming data from usually limited plans. We have developed a passive monitor, Adkintun mobile[2] , which gathers information from Chilean companies. The information collected so far has given us a good approximation to the network state in respect of the quality of the link between terminal and antenna, but it misses events which occur in high-level APIs.

With all the projected data collected we can help to improve how mobile software can react to changes on the network quality and even on the perception of the user of the network quality.

In order to achieve this goal, we propose our design and future work towards an architecture which uses network and perception metrics to contextualise the running application. It will allow mobile applications to adapt themselves on faulty network conditions and enhance the user experience.

The rest of the paper is structured as follows. In Section II we describe what kind of perception metrics we need to measure. In Section III we present our requirement and challenges to measure mobile internet quality on mobile devices. Then in Section IV we propose an architecture which gathers and process the metrics to obtain statistics and provide information to the application. Finally in Section V we present our further work planning.

[1]http://www.businessweek.com/news/2012-10-29/google-says-700-000-applications-available-for-android-devices

[2]http://www.adkintunmobile.cl

II. Quality perception of network state

Mobile software is usually intended for end-users. So in addition to be functional and efficient, it has to be always responsive in order to be it attractive and frustration-free for the user. Research has been done in the field of Quality of Experience [12] in order to help to develop usable software. This helps designers and programmers provide better ways to show information and interact correctly with the user.

Research normally focuses in analysing an improving user interactions at development time. That is, gathering information at design phases to determine the best user interfaces and interactions. It is also possible to collect usage information from running instances, however they are usually used only to enhance subsequent versions.

We propose to employ the runtime user interactions to reveal software quality metrics. In particular we are interested in discover user discomfort produced by faulty mobile internet connection. This is because there is a relationship between Quality of Experience and Quality of Service [5], [9]. For instance, when user repeatedly presses the reload button it can mean the application is not receiving information at the usual pace and it should perform adaptations to make the view more responsive. Adaptations can be conducted in different ways, showing more information or explaining the possible problems, adjusting terminal's settings, or even switching to more lightweight algorithms to deliver the important information as quickly as possible by degrading other functions.

In order to better understand and measure the user's discomfort regarding the network state we need to perform a number of tasks. First, we need to study how a faulty network can affect the responsiveness of the final application. We first propose to study the OSI model [13]. We want to examine each one of the OSI layers to determine how each one of them can impact the perceived software quality, for instance because of high latency, slow start-up or slow transfers. This study will guide us to better link how network issues affect how users perceive the software's responsiveness.

We target to develop a perception model to allow us collect information about the runtime interactions and determine when applications need to optimise themselves to provide better user experience. This model should define several metrics to measure user perception of quality and they need to be designed in such ways they will reveal network issues and not other source of problems. We also need to propose a methodology to measure interactions, and possibly define a domain-specific language to define observation points. Finally we need to design how this model will represent and transmit the information given we target to mobile devices, which are limited in terms of battery life, storage capacity and network usage.

III. Measuring mobile networks

Several techniques exist to determine the network state [14], [15], [16]. Most of them consider actively using the machine's communication facilities API or access to non-public information to perform measurement tasks contacting external servers. We have previously stated [11] we cannot rely in such techniques since mobile internet plans provide a limited number of megabytes before the connection limits its speed or more expensive per-session fees are charged to the user's account.

For this task we have developed Adkintun mobile, a passive monitor which runs on Android devices. This monitor collects information about the connected cellular antennas, type of connection and average download/upload speed rates from over 200 volunteers in Chile. Statistics will be available during the first quarter of 2013. This approach allows us to measure the network state according to the quality of the connection to the cellular antennas. It also avoids consuming bytes from the mobile internet plan since the information is gathered offline and stored in the phone until a WiFi connection is found and used to transmit the information.

Although we have successfully obtained the information from Android devices, in other platforms we have found limitations in the API permissions to access low-level mobile network information. In addition, we are not measuring high-level network status since we measure global device statistics and we cannot observe other application's network connections in a legit way.

Given the current limitations and needs, we have to define new techniques to measure network status with the inherent limitations of mobile devices.

A. Semi-active measuring

In order to obtain more metrics of the network's state we need to define a technique to gather relevant information about high-level network status. For this approach we have the following constrains:

No standalone application : The user should not be forced to open a specialised application to measure its internet quality.

Authorised APIs : The implementation must be accepted in official marketplaces.

Small fingerprint : The data must be gathered from the interaction of applications with the network APIs and not from traffic generated by measure algorithms.

Given the constraints listed above, we propose to add a layer on top of the network APIs in order to provide networking services and also gather statistics about the quality of the internet connection. The information obtained, as in the passive monitor, is sent to the central server only when the application is connected to a WiFi interface, thus avoiding the use of the mobile internet plan. We will adapt classic metrics such as speed testing, dns lookup time, and

latency to perform them with minimun impact in the mobile terminals. We also plan to add other metrics such as network errors and repetitive network switching to our metrics set.

We consider that all gathered information is also interesting to the application using this layer, since it can use the network's state data to implement algorithms to cope with poor connectivity scenarios, hence proving a more graceful degradation of its services. In order to allow this kind of adaptation we need an architecture to be able to manage all the gathered information to correctly infer the state of the connection. Then we need to reveal in consistent and coordinated means the state of the network to the running application to let it properly adapt itself. We discuss in the next section an architecture which integrates our metrics and how can be used to develop such kind of adaptations.

IV. An architecture to enhance mobile user experience

In the previous sections we have discussed our goals to measure both network and perception metrics related to the network-related user experience. We now present an architecture to integrate all the gathered information. In order to provide relevant context to the running application we define the schema shown in Figure 1. This is divided in four main modules: network sensor, perception sensor, context provider and runtime adaptations.

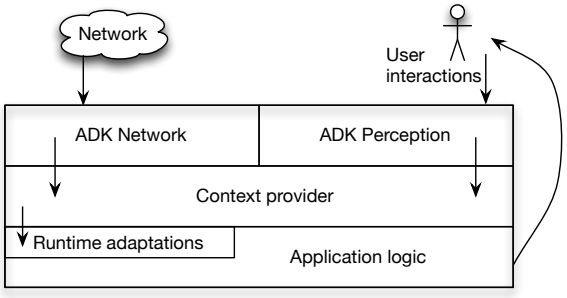

Figure 1. Architecture to enhance mobile use experience

A. Perception sensor

This sensor, defined in Section II is in charge to measure and manage the metrics of user perception of the quality of the network connection. This information is sent to the context provider which also receives data from the network sensor.

B. Network sensor

We have defined in Section III the need for mobile device oriented network metrics. This module should implement those metrics, which are optimised for mobile devices. In this case, act as a networking layer, which in addition to provide connectivity with external hosts, it will gather statistics of active connections. All this data is sent to the context provider to allow it expose the current state of the network to the running application.

C. Context provider

This module acts as a receiver of all the sensed information and it is in charge of processing and inferring relevant data to determine what is the state of the network. This layer also has to implement correct means to reveal the information to the running application. This can be performed by different means and further research is needed to determine which is the best way to represent it.

Once we can determine and reveal the current network state, the application should be able to react to that state. This can be done by different means, from very basic strategies to more elaborated and modern techniques. We describe some approaches in the next subsection.

D. Runtime adaptations

Given that applications are aware about the network state, they can adapt themselves to better perform on different conditions. The most basic approach is using if-statements to determine the state in a given time and perform accordingly, however this is discouraged since this will pollute the running code with adaptation conditions.

Traditional object-oriented techniques such as the strategy pattern allow to modularise the different implementations in separated components, but they still need to introduce conditions to setup the adaptable sections.

Aspect-oriented programming (AOP) [17] is proposed to modularise crosscutting concerns. Its pointcut-advice model allows the programmer define points in the code which are intercepted and can manipulate their behaviour at runtime. A drawback for runtime adaptations is that AOP separates the different variation of the algorithms in separated modules (aspects), even when adaptations are in fact part of the main features.

Context-oriented programming (COP) [18] is another paradigm to allow applications to perform runtime adaptations. COP considers context-oriented adaptations as part of the main application concerns and define that adaptation constructs in the same module. Then, in runtime the application will dynamically select the best implementation regarding the current context. Implementation of COP in mobile languages include ContextJ [19], Flute [20] and Subjective-C [21].

V. Conclusions and future work

Mobile software development raises concerns not found in desktop and server computing: limited resources, an ever changing environment, need for personalised experience and demand of reliable and constant internet connection. Mobile software is a market which getting more attractive every year and hence the need to cope with these new challenges to provide better applications.

We propose an architecture which would help users and developers to improve the services provided by mobile application in regard to the mobile network quality. In one hand, we gather information about the connection quality associated with its location by using network and perception metrics. In the other hand we can provide the running application with valuable information to allow them adapt and improve how it interacts with the user.

We plan to follow this research splitting it in 3 aspects:

1) User's quality perception metrics for mobile internet.
2) Mobile internet quality metrics.
3) Network state and QoE context provider.

State of the art runtime adaptation mechanisms will be used in running examples, but their study is out of the scope of our research objectives.

Further publications will be focused in each one of our aspects separately with the main goal of obtaining a system to allow applications to self-adapt in face of different network conditions.

ACKNOWLEDGMENTS

This work is part of Regional Program STIC-AmSud-CONICYT and is funded by National Programs INNOVA CORFO (Chile) 12IDL1-15665 and 12IDL1-16017. We would like to thank Javier Bustos and Javier Pereira for their support towards our work.

REFERENCES

[1] Canalys, "Smart phones overtake client PCs in 2011," Canalys, Tech. Rep., 2012, available at http://www.canalys.com/newsroom/smart-phones-overtake-client-pcs-2011.

[2] N. Wood, "Mobile data traffic growth 10 times faster than fixed over next five years," Sep 2009, available at http://www.totaltele.com/view.aspx?ID=448681.

[3] comScore, "2012 Mobile Future in Focus," 2012, available at http://www.comscore.com/Press_Events/Presentations_Whitepapers/2012/2012_Mobile_Future_in_Focus.

[4] I. O. for Standardization, "Iso 9241-210:2010 ergonomics of human-system interaction – part 210: Human-centred design for interactive systems," Directly by the International Organization for Standardization, 2010. [Online]. Available: http://www.iso.org/iso/iso_catalogue/catalogue_ics/catalogue_detail_ics.htm?csnumber=52075

[5] M. Koivisto and A. Urbaczewski, "The relationship between quality of service perceived and delivered in mobile internet communications," *Inf. Syst. E-Business Management*, vol. 2, no. 4, pp. 309–323, 2004.

[6] R. Hofman, "Software quality perception," in *SCSS (2)*, K. M. Elleithy, Ed. Springer, 2008, pp. 31–37.

[7] ISO/IEC, *ISO/IEC 9126. Software engineering – Product quality*. ISO/IEC, 2001.

[8] D. Collange and J.-L. Costeux, "Passive estimation of quality of experience," *Journal of Universal Computer Science*, vol. 14, no. 5, pp. 625–641, mar 2008, http://www.jucs.org/jucs_14_5/passive_estimation_of_quality.

[9] S. Khirman and P. Henriksen, "Relationship Between Quality-of-Service and Quality-of-Experience for Public Internet Service," in *PAM 2002, Passive and Active Network Measurement workshop*, Mar. 2002.

[10] J. Shaikh, M. Fiedler, and D. Collange, "Quality of experience from user and network perspectives," *Annales des Télécommunications*, vol. 65, no. 1-2, pp. 47–57, 2010.

[11] G. Hourton, G. D. Canto, J. Bustos, and F. Lalanne, "Crowd-measuring: Measuring quality of mobile internet from mobile terminals," in *6th International conference on networks games, control and optimization (NetGCooP)*, 2012.

[12] M. Singh, "U-scrum: An agile methodology for promoting usability," in *Agile, 2008. AGILE '08. Conference*, aug. 2008, pp. 555 –560.

[13] H. Zimmermann, "Osi reference model–the iso model of architecture for open systems interconnection," *Communications, IEEE Transactions on*, vol. 28, no. 4, pp. 425 – 432, apr 1980.

[14] H. Falaki, D. Lymberopoulos, R. Mahajan, S. Kandula, and D. Estrin, "A first look at traffic on smartphones," in *Proceedings of the 10th ACM SIGCOMM conference on Internet measurement*, ser. IMC '10. New York, NY, USA: ACM, 2010, pp. 281–287.

[15] Ookla, "Speedtest," http://speedtest.net/mobile.php.

[16] M. Research, "Mobiperf," http://mobiperf.com.

[17] G. Kiczales, "Aspect-oriented programming," in *ICSE*, G.-C. Roman, W. G. Griswold, and B. Nuseibeh, Eds. ACM, 2005, p. 730.

[18] R. Hirschfeld, P. Costanza, and O. Nierstrasz, "Context-oriented programming," *Journal of Object Technology, March-April 2008, ETH Zurich*, vol. 7, no. 3, pp. 125–151, 2008.

[19] M. Appeltauer, R. Hirschfeld, M. Haupt, and H. Masuhara, "Contextj: Context-oriented programming with java," *Information and Media Technologies*, vol. 6, no. 2, pp. 399–419, 2011. [Online]. Available: http://ci.nii.ac.jp/naid/130000770579/en/

[20] E. Bainomugisha, J. Vallejos, C. De Roover, A. L. Carreton, and W. De Meuter, "Interruptible context-dependent executions: a fresh look at programming context-aware applications," in *Proceedings of the ACM international symposium on New ideas, new paradigms, and reflections on programming and software*, ser. Onward! '12. New York, NY, USA: ACM, 2012, pp. 67–84.

[21] S. González, N. Cardozo, K. Mens, A. Cádiz, J.-C. Libbrecht, and J. Goffaux, "Subjective-c: Bringing context to mobile platform programming," in *SLE'10: Proceedings of the Third international conference on Software Language Engineering (SLE 2010)*, ser. Lecture Notes in Computer Science, no. 6563. Berlin, Heidelberg: Springer-Verlag, 2010, p. 246265.