Crowdsensing Mobile Content and Context Data: Lessons Learned in the Wild

(Invited Paper)

Katia Jaffrès-Runser, Gentian Jakllari, Tao Peng, Vlad Nitu
Institut de Recherche en Informatique de Toulouse
Université de Toulouse, INPT-ENSEEIHT, Toulouse, France
Email: {kjr, gentian.jakllari, tao.peng, vlad.nitu}@enseeiht.fr

***Abstract*—This paper discusses the design and development efforts made to collect data using an opportunistic crowdsensing mobile application. Relevant issues are underlined, and solutions proposed within the CHIST-ERA Macaco project for the specifics of collecting fine-grained content and context data are highlighted. Global statistics on the data gathered for over a year of collection show its quality: Macaco data provides a long-term and fine-grained sampling of the user behavior and network usage that is relevant to model and analyse for future content and context-aware networking developments.**

I. Introduction

This paper underlines the main takeaways and design choices made for the deployment of a crowdsensing application in the context of the CHIST-ERA Macaco project. The goal of this project is to leverage networking content and context to design novel post-4G mobile offloading mechanisms. The project aims at providing seamless and cost-efficient connectivity to mobile users by adapting data transport decisions to the near future, and thus leveraging past context changes and data requests. Such adaptive learning mechanisms rely on the quality of a set of continuously collected data samples.

In the Macaco project, several reasons led to the deployment of an opportunistic mobile crowdsensing app[1] that periodically samples the so-called networking context and content. The first reason is the need for collecting knowledge on users networking habits at the early stage of the project. From such collected data, user behavior can be analyzed, understood and modeled by deriving different user behavior profiles. Another fundamental reason is that the final learning mechanisms will rely on sensing functions that store content and context history either on users mobile phones or on a remote server. As such, final Macaco algorithms can benefit from the sensing capabilities developed for the initial crowdsensing app.

This paper introduces the issues related to sensing context and content data on a mobile platform in Section II. The hardware and software architecture of the Macaco sensing app is presented in Section III. This part highlights the lessons learned from the Macaco app design and deployment stages, mainly dealing with energy-efficiency. Global statistics on the data collected in the project are underlined in Section IV and finally, Section V concludes the paper.

[1] http://macaco.inria.fr/macacoapp/

TABLE I
Content and context samples sensed in Macaco app.

Context	WiFi scan	3G scan	Bluetooth scan	IP address
	Location	Acceleration	Memory	Battery level
Content	URL	App name	App upload	App download

II. Crowdsensing context and content

Networking context defines in our case the environment the mobile user is experiencing. Context can be described by various pieces of information such as user location, user motion or specifics on the currently available wireless networks. It is obtained by triggering system calls to various sensors (e.g. GPS, accelerometer, gyroscope, battery level, etc.) or to network interfaces (e.g. WiFi, Bluetooth, Cellular, etc.). Context data is usually easy to retrieve, but, unfortunately, related system calls are often energy consuming. As such, the periodic collection of context data has to be engineered in an energy-saving manner. The energy-efficient sensing strategies developed for Macaco are discussed in Section III-B.

Networking content relates to the nature of the data that is either pushed or pulled from or to the Internet by the mobile user. Content can be described by more or less precise pieces of information, going from the accurate description of the data (e.g. the file transfered contains episode 1 of Star Wars) to the raw information where you only know that content is an audio file, for instance. Crowdsensing content on mobile devices is not an easy task as it depends on the privacy rules enforced by the mobile platform operating system. Best case scenario is to have access to the full content exchanged by a mobile user with Internet. Unfortunately, complete access is unrealistic for several reasons. Most of the content is manipulated inside mobile applications that can't be accessed by third party applications without their explicit authorization. The operating system may have access to raw data inside the networking stack (provided that the data exchanged isn't ciphered) but on-the-fly content analysis is computationally expensive and requires superuser privileges. As such, networking content is described with less precise but still informative meta-data.

Meaningful and relatively easy to access meta-data is given by the URLs (Uniform Resource Locator) or URIs (Unified Resource Identifiers) of contents being requested by users. Another interesting but less precise piece of information is

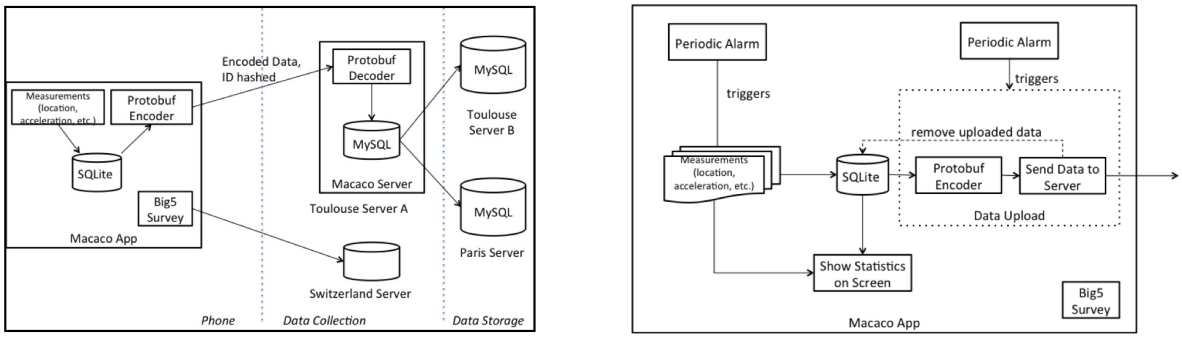

Fig. 1. Overall Macaco crowdsensing architecture (left) and Macaco app architecture (right)

given by simply tracking the name of the mobile app that has generated traffic in a given time interval. Exact content of the retrieved data isn't known, but it is possible to provide a coarse classification of content based on such information. For example, a video streaming application that triggers a large amount of data download obviously calls for a video download operation. Exact video name isn't known, but the fact that a video was downloaded can be leveraged to understand user networking habits. Macaco sensing app is retrieving both types of content meta-data. URLs are extracted from the default browser history changes. We collect as well the name of the apps that have pushed or received bytes to/from the Internet since last sensing operation. Table I enlists the different content and context samples sensed in our app.

III. Design of a crowdsensing mobile app

At design stage, one should bear in mind that the quality and quantity of data collected is strongly conditioned by the motivation of the sensing participants [1]. The key conditions for having a sensing app adopted for a long time by participants are the following ones:

- the sensing app shouldn't disturb regular device's run,
- it shouldn't put too much stress on the overall energy consumption,
- upload data to the collection server at no financial cost,
- data collected should guaranty privacy to participants.

Moreover, if the collection requires the data to be collected from the largest group of participants possible, a good incentive mechanism has to be created. Trading participating efforts for money can not easily scale to large groups. The most successful collection is obtained if the app offers an added service that becomes the main reason for users to participate. Finding the service that motivates users is difficult, and from our experience, should be seen as one of the key issues to be addressed at early design stages.

A. The Macaco architecture

The overall architecture of the Macaco crowdsensing app is represented in Fig. 1-(left). All Macaco sensing apps push collected data to front-end servers. The one located in Toulouse collects the periodic samples of all data enlisted in Table I. The server in Switzerland collects the answers of a personality test that participants take once based on the state-of-the-art Big Five psychology test [6]. Data is stored in a MySql database on the front end server. The data collected each day is sent over night to two storage servers located in Toulouse and Paris. All data sent from the mobile phones to the front-end servers are only done once connected to WiFi to avoid costs. The Macaco app sends the data using an energy efficient serialization library provided by Android.

To warrant the best privacy practices to our participants, we have followed the privacy enforcement rules of CNIL (Commission Nationale de l'Informatique et des Libertés), the French privacy regulation body. Thus, all samples sent by a mobile phone are identified on our server with a SHA-256 hash of the mobile IMEI (International Mobile Equipment Identity). Communications between servers use secure communications and access to stored data is only possible for identified Macaco project members. Moreover, data will only be stored for a limited duration in a non-anonymized state.

Long-time adoption of a crowdsensing application by participants necessitates a careful mobile app software design. In our implementation, we focus on Android mobile phones with minimum Android API 16 (i.e. Android 4.1-Jelly Bean). This choice was made to target the most widely used Android version in 2014. The app was designed such as to provide a periodic measurement of the data of Table I. The mobile app architecture is given in Fig. 1-(right) and runs as a foreground Android service. Two periodic alarms are implemented, one for triggering the data collection and the other one for pushing data to the front-end servers. Data collection period is set to 5 minutes. Data upload period is set to one hour, but it is conditioned on the mobile phone being connected to Internet over WiFi. Collected data is temporarily stored in a SQLlite database. This data is analyzed on the phone to show simple statistics on the sampled data to the participant on demand.

B. Energy efficiency

The most challenging part in a crowdsensing app design is to minimize energy consumption. It is possible to measure the app's energy profile using specific monitoring hardware and software [2]. Using their energy profiling platform, authors of [2] have observed for the Macaco app that the highest energy consumer tasks are GPS localisation [$\sim$6Wh], followed

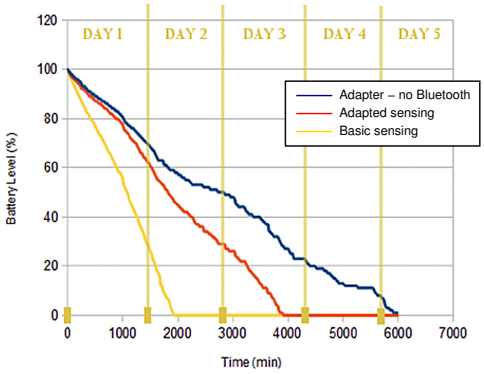

Fig. 2. Impact of energy-related optimization on context data retrieval

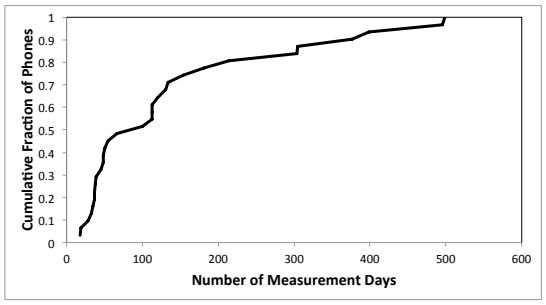

Fig. 3. Number of days Macaco collected measurements.

by Bluetooth, accelerometer, and Wi-Fi scanning [$\sim$0.4Wh each]. Following, we present the main solutions implemented to reduce energy consumption of network scanning operations and of location retrieval.

Network sensing: WiFi and Bluetooth sensing tasks are energy hungry tasks because the physical layer has to scan all channels for beacons and retrieve all information on available networks. At each sampling stage, you can ask for a new scan, but it is more energy efficient to simply poll the Android API to get the lasted scan results that are populated periodically by the networking interface of Android.

Location sensing: The most precise location sensing is obtained with a GPS service call. However, GPS may fail if not enough satellites are visible (this is often the case indoor). As such, the naive sampling method that calls GPS updates every five minutes is energy hungry and not reliable. Android offers an alternative localisation service in its API that obtains user's location from an in-house localization protocol that requires Internet access to query a remote server. Moreover, Android stores the *last known location* of the user and offers it to any app requesting it. This last know location is populated by any location request made (either using GPS or network request) by any app running on the phone. To take advantage of both features, we call the API for network location update by sampling the *last known location* global variable of Android. Then, to enforce a precise location in this *last known location* variable, we check whether user has changed its location or not. If the user has moved, we call the GPS to get a precise location. Deciding whether the user has moved follows a simple heuristic: if the strongest three access points returned by WiFi scan have changed, we trigger GPS localization. If not, we assume the user hasn't moved.

Energy consumption of a single stationary device (MotoG 1st generation) has been tracked in Fig. 2. The battery depletion is compared when (i) location sensing is made every 5 minutes (basic), (ii) location sensing is made only when user motion is detected (adapted sensing) and when (iii) adapted sensing is made without Bluetooth scan. Battery life is extended by 100% if location is more carefully sampled. Moreover, stopping Bluetooth improves again lifetime for more than a day.

IV. Results

In this section we present some preliminary results from the deployment of Macaco by 21 mobile users selected among students and workers. Part of them were incentivized financially by reimbursing their data plan fee. The data is organized in three groups. First, we present high-level data on the deployment of Macaco. Second, we present statistics on the volume of mobile data. Finally, we provide detailed statistics on how users interact with the applications installed on their phones.

A. Statistics on participation

Fig. 3 shows the number of days for which Macaco reported measurements from 21 participants in the experiments. The data shows a high variability in the number of days Macaco has been running user' phones, explained by the fact that volunteers are added all the time. However, it is interesting to note that for about half the users Macaco has collected data for periods between 100 and 500 days.

B. 3G/WiFi usage

In this section we present Macaco measurements regarding the data traffic on the mobile phones. In particular, we are interested in knowing the amount of traffic transmitted over the cellular network and WiFi, respectively. This kind of data could, for example, help cellular providers customize their data plans and from the user's perspective, quantify the potential for off-loading. The data in Fig. 4 leads to some interesting observations. First, user traffic is highly variable, with some

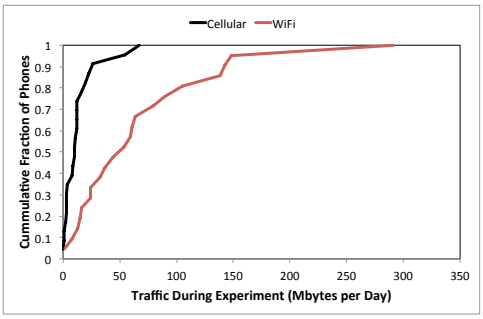

Fig. 4. Traffic observed during the measurement period.

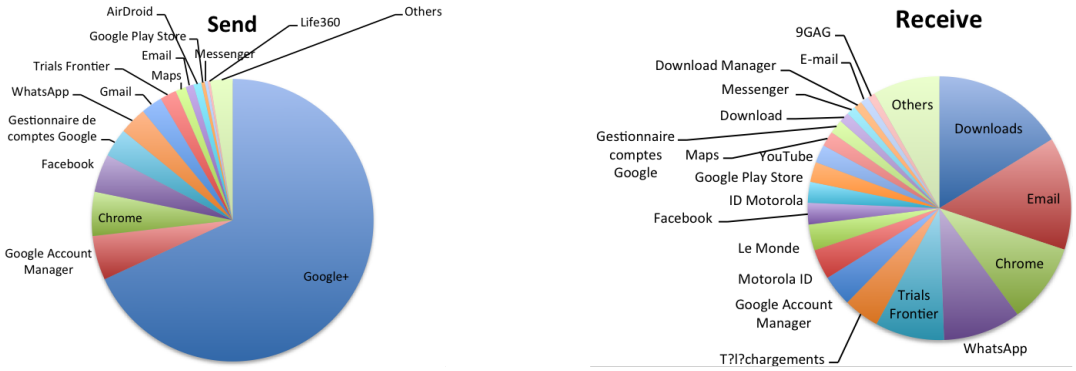

Fig. 5. Pie chart of traffic volume generated in upload (Send) and download (Receive) between July 2014 and March 2015. Total volume in upload is 10551MBytes and in download is 3798MBytes.

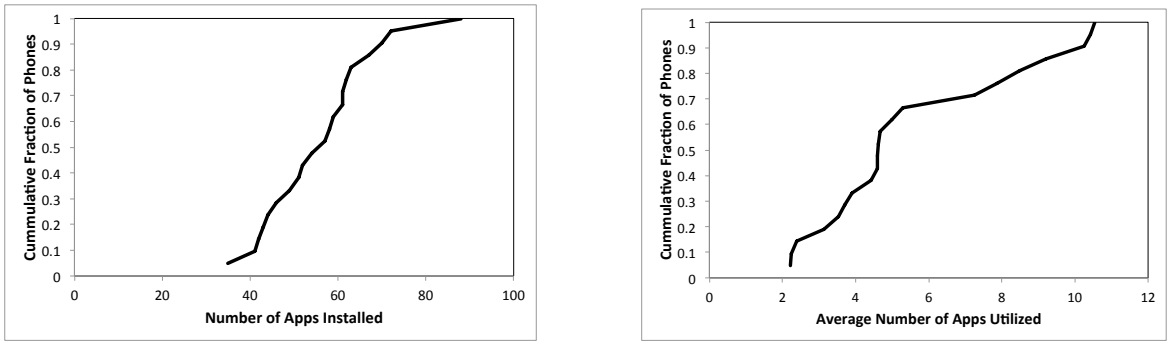

Fig. 6. Number of Apps installed on Macaco users' phones (left) and average number of Apps generating traffic during the measurement periods (right).

users generating no traffic at all while others generating several dozens Mbytes per day. Second, WiFi traffic is significantly higher than the cellular traffic. While the median user generates around 10 Mbytes per day on the cellular network, it generates around 50 Mbytes per day on WiFi, a 5-fold increase. This could be due to the ubiquitous presence of WiFi access points, or the judicious usage of the data plans of the users running Macaco, or both. Finally, the data shows the presence of the well-known power users [5]: around 5-10% of users generate a very high amount of traffic both on the cellular network, between 40-60 Mbytes per day, and WiFi, between 150-290 Mbytes per day.

C. Mobile apps statistics

In this section, we turn our attention to content and present Macaco measurements regarding the application usage of the participants in the study. First, we were interested in knowing the volume of applications installed and actually generating traffic. Fig. 6-(left) shows that users in this study install a large number of applications, with the median users installing about 55 applications. However, looking into how many applications actually generate traffic, the numbers look very different. Fig. 6-(right) shows that only a small fraction of the applications installed are actually used. The median participant uses only about 5-6 applications.

Finally, we were interested in identifying the most important applications, that is the applications that generate the most traffic. This information can be very useful to pre-fetching solutions [4], among others. Fig. 6 shows the amount of data traffic generated by every application across all the Macaco participants. The data is divided into upload (send) traffic on the left and download (receive) traffic on the right. The first thing the data shows is that upload and download traffic do not follow the same pattern. Google+ dominates the upload traffic while the download traffic pie is cut more uniformly. E-mail, Chrome and WhatsApp dominate the download traffic. Furthermore, the data shows that over 20 applications contribute significantly to the traffic users generate. Considering that, as per the data of Fig. 6-(right), participants use around 5-6 applications, on average, this means the most popular applications vary from user to user. Therefore, careful consideration must be paid by solutions targeted at pre-fetching popular applications.

V. Conclusion

This paper has presented the main take-aways from a crowdsensing experiment where mobile content and context data was collected. New open frameworks exist for crowdsensing mobile phones [3] that should be compared to in-house solutions. Moreover, recent Android releases (starting Android 5.0) provide an energy-efficient tasks scheduler that could be leveraged for our crowdsensing app. The data collected through Macaco experiment is really valuable as it has succeeded in capturing the long-term and fine-grained networking usage of mobile users.

ACKNOWLEDGMENT

This work was supported by CHIST-ERA Macaco project, ANR-13-CHR2-0002-06

REFERENCES

[1] G. Chatzimilioudis, A. Konstantinidis, C. Laoudias, and D. Zeinalipour-Yazti. Crowdsourcing with smartphones. *IEEE Internet Computing*, 16(5):36–44, Sept 2012.

[2] A. Ferrari, D. Gallucci, D. Puccinelli, and S. Giordano. Detecting energy leaks in android app with poem. In *2015 IEEE International Conference on Pervasive Computing and Communication Workshops (PerCom Workshops)*, pages 421–426, March 2015.

[3] Nicolas Haderer, Romain Rouvoy, and Lionel Seinturier. Dynamic Deployment of Sensing Experiments in the Wild Using Smartphones. In François Taïani and Jim Dowling, editors, *13th International IFIP Conference on Distributed Applications and Interoperable Systems (DAIS)*, volume 7891, pages 43–56, Firenze, Italy, June 2013. Springer.

[4] Brett D. Higgins, Jason Flinn, T. J. Giuli, Brian Noble, Christopher Peplin, and David Watson. Informed mobile prefetching. In *Proceedings of the 10th International Conference on Mobile Systems, Applications, and Services*, MobiSys '12, pages 155–168, New York, NY, USA, 2012. ACM.

[5] U. Paul, A. P. Subramanian, M. M. Buddhikot, and S. R. Das. Understanding traffic dynamics in cellular data networks. In *2011 Proceedings IEEE INFOCOM*, pages 882–890, April 2011.

[6] L. Pervin. *Handbook of Personality: Theory and Research*. Guilford Press, 1990.