Beyond San Fancisco Cabs: Building a *-lity Mining Dataset for Social Traces Analysis

Marc-Olivier Killijian, Matthieu Roy, Gilles Tredan

Beyond San Fancisco Cabs : Building a *-lity Mining Dataset for Social Traces Analysis

Marc-Olivier Killijian Matthieu Roy
CNRS; LAAS
Université de Toulouse ; UPS , INSA , INP, ISAE
F-31077 Toulouse, France
{marco.killijian| roy }@laas.fr

Gilles Trédan
TU Berlin
Berlin, Germany
gilles@net.t-labs.tu-berlin.de

ABSTRACT

In this paper, we report our advances, choices and first insights in the design of a mobile phone powered data collection platform. We believe that collecting such data is vital to achieve a better understanding/modeling of several phenomenons related to human activity (e.g. mobility, social contacts, or terminal failures). However, designing such a platform raised a lot of questions discussed in this paper.

1. MOTIVATION

Context.

The ubiquity of geo-located devices (mobile/smartphones, GPS, etc.) permitted scientists to collect datasets containing mobility information. Mining reality (and mobility) out of these datasets has drawn a lot of attention, for example for studying the spread of viruses [12, 9], for designing socially-aware network routing protocols [8, 2], and in the raising dynamical networks community. These datasets can also be used to design models of users behaviors[5, 7, 17], phone usage, etc.

Background.

We are interested in building *resilient ubiquitous mobile systems*. To attain this goal, several aspects of these systems need to be investigated, from their formal foundations to their experimental evaluation. As such, we worked on distributed [16] and cooperative [10] algorithms for such systems, but also on both analytical [3] and experimental resilience evaluation [11] of mobile systems. One of the major challenge when modeling a mobile distributed system is to provide an adequate mobility and connectivity scheme, that should represent, as precisely as possible, actual interactions between *human beings*. Mobile devices are the first sensing devices that are almost coupled with their carrier, allowing to capture human activity with an unprecedented resolution. This drove us to dig in research on mobility datasets, to perform experimental and analytical evaluation based on actual mobility traces.

Indeed, as far as analytical evaluation is concerned, we needed to use parameters representing the rate at which a node encounters a cooperating peer, the rate at which it gets Internet connectivity, the rate at which it (or any peer) fails, etc. We had to choose these parameters (and their distribution law) by rule of thumb but definitely wanted to get actual and accurate data to backup these estimates. Regarding experimental evaluation, we are building an emulation platform based on small robots that carry laptops running the algorithms. At the moment we used predefined vehicular scenarios but we want the platform to be able to use proper mobility models.

Mobility and Social Models.

It is worth noticing that the whole mobile systems community struggles with classical random-* mobility models. They are too far from the application scenarios to be sound. Recent research works try to tackle this problem (e.g., in [13]). Yet, there is still a need for extending mobility models with failure models for small devices such as mobile phones. Indeed, as these devices are mass-produced and carried by users, they are more prone to failures: they move a lot, are light, and henceforth are prone to fall; they have a small battery and are prone to energy depletion; they are used for gaming or multimedia purpose, users install a bunch of applications and henceforth they are prone to software failures. For all these reasons, an ideal dataset would include information about energy consumption, operating system reboots, mobility, etc.

To the best of our knowledge, there is no dataset that satisfies this specification. Most of CRAWDAD[1] datasets were produced in a limited vicinity, either during a conference, or on a campus. Very few available datasets include precise localization data [14], most of them focused on contact traces. Unfortunately phone operators are not keen to release their data and do so

[1] `http://crawdad.cs.dartmouth.edu/`

to only some lucky few[7, 17]. Furthermore, there are very few data collection tools available. For example, the software used to build the Reality Mining dataset [4] is not maintained anymore [15]. This drove us to decide to build our own platform for collecting mobility, failure and energy consumption data.

2. A PLATFORM FOR COLLECTION OF MOBILITY AND SOCIAL TRACES

2.1 Operating System Platform

Recently, mobile phone platforms have drawn a lot of attention from manufacturers and users, mainly due to the opportunity to embark what has become lightweight computers in everyone's pocket. As such, current mid- to high-end phones are now equipped with large screens, GPS, high speed cellular network access, local wireless network interfaces, and various sensors (accelerometers, magnetometers, etc). As for every computing device, interaction with hardware is done through an operating system, using an API. The different phone manufacturers offer platforms that differ radically in their operating system and API choices. In this section, we describe the various options available, and explain which platform we chose and why such a choice.

Indeed, from our point of view, the rationale to choose one platform instead of another is drawn by two main reasons:

- the platform should offer a simple way to access as much important parameters on user's activity as possible,
- the platform should be desirable for users, so that users will accept to use the platform as their main phone, and run our software

In fact, as we shall see, these two goals are a bit antagonist, since the most desirable phone platform for users to date, the Apple iPhone, is also the most closed platform from a programmer's point of view.

Most mid- to high-end phone on the market now provide a GPS for mobility tracing, at least one wireless network interface (bluetooth), and a large touchscreen. The difference between available platforms resides in their operating system and their development model (API and software distribution model). There are now five main phone platforms: Apple iPhone, Symbian Foundation, Google Android, Microsoft Windows Mobile, and the newly released joint system from Nokia and Intel, MeeGo.

Apple iPhone.

Apple platform for mobile phone is one of the most impressive successes in recent years. It is considered appealing from the user experience point of view. Yet, the development model is strongly tight to Apple, since it is not possible to install applications on the phone without passing through Apple certification program. Moreover, Apple engineers made some radical choices to guarantee a high reactivity of the platform, the most stringent from a programmer's point of view being its inability to run more than one non-Apple application. It is thus currently impossible to run a dedicated background task that analyzes user activity.

MeeGo.

MeeGo is a new platform, based on the fusion of two linux-based platforms, Intel Moblin and Nokia Maemo. From a developer's point of view, this platform seems an optimal choice : it is open (source code available), it is truly community-based (maemo has been developed on a community-based model from 2005), and it is based on linux. For the moment, available hardware platforms that run this system are rare (only one phone), and rather expensive. For us, it seemed that this platform is not mature enough to use it for real user deployment.

Google Android.

Google released the first version of Android software at the end of 2007. This platform is thus still young, but has major benefits: it is a true linux-based UNIX system, it is open, and it is strongly pushed by Google, which should result in a future commercial success. Unfortunately, our preliminary tests showed some drawbacks of the platform, such as low battery life, instability, and the lack of a cheap android-compatible phone, resulting in a relatively small user base.

Microsoft Windows Mobile & RIM.

Research In Motion's BlackBerry and Microsoft Windows Mobile are two message-oriented systems. They are widely used in businesses, and have been traditionally closed systems. Recently, due to pressure from other systems, RIM has developed and application store, and released an API for java programming, and Microsoft announced its new system, optimized for touch interfaces. Yet, the high price of terminals and the relatively low programmers community led us to use another platform.

Openmoko.

Openmoko is a linux-based phone platform, that provides an open source software stack. The project is still in development phase, and not ready for general public resease.

Symbian.

Symbian used to be a proprietary operating system used by various manufacturers. In 2009, its license scheme has changed to open source. Available on a wide

variety of terminals, tt is currently the major operating system for phones (46.9% worldwide market share in 2009[2]). Recently released as open source, development tools are easily available, and not being tight to a manufacturer, it is possible to develop an application without having to pass it through manufacturer certification process. Symbian community is actively updating it, with programmed releases up to at least 2011.

Conclusion.

From these characteristics, the best platform for performing a background collection of various parameters on a large base of users appeared to us to be the Symbian platform: information for programming is easily available, development tools are mature, the system is very stable, it can run multiple applications in parallel, and it is actively developed and maintained.

2.2 Hardware Platform

We want to provide many users with smartphones equipped with our software. This led us to find a phone (1) that can *sense* many parameters (at least GPS, WiFi, bluetooth), (2) that has a good autonomy to support the additional energy consumption necessary for our logging application, (3) that is cheap and, last but not least, (4) that is considered desirable for users.

We finally opted for the Nokia 5800 smartphone. Its retail cost is about 230€, which is relatively cheap when considering that it provides all hardware we needed, a GPS chip, a large (640∗360) touchscreen, an accelerometer, a compass, a WiFi interface and a bluetooth interface in a small form factor (about $100g$). Moreover, this phone is considered a good one from the users point of view, due to its light weight, its good camera, its free navigation system and the fact that it can be used with video conference softwares such as Skype.

2.3 Design: what can be sensed

When designing our application, it appeared that the choice of the programming language would have an impact on available data that can be sensed. In Java (be it Java Micro Edition or Java Standard Edition), no platform allows a program to get access to low-level parameter such as the list of available WiFi interfaces. Thus, we had to use system-oriented API and programming language. Symbian standard interface is a C++ API that gives the programmer access to most of the capabilities of the phone. In our case, we listed the following interesting sources of information:

- GPS information. Location information is essential when building a mobility trace. This piece of information will be sampled at fixed interval. It can also generate events when passing near some predefined interest points.
- WiFi access points, and WiFi usage.
- Cellular information.
- Nearby bluetooth devices.
- Battery level. This should be sampled, and the logging software should collect power events (charging, battery warning) when they occur.
- Phone calls.
- Accelerometer. Although this parameter is not *per se* related to interactions or mobility, it may be used to detect inactivity.
- Compass.
- Proximity Sensor senses wether the surface of the phone is close to an object (a pocket, an ear)
- Light Sensor.
- Power consumption related resources: CPU usage, current intensity drawn from the battery.

[2]source: Gartner inc.

2.4 Implementation considerations

In our first implementation, we limited the logging to a subset of available sources of information, as summarized in Figure 1:

Information source	**periodicity**	**fixed/variable size**
GPS	periodic	fixed
WiFi	periodic	variable
Bluetooth	periodic	variable
Battery level	periodic	fixed
Battery events	sporadic	fixed
Phone calls	sporadic	fixed
Reboots	sporadic	fixed

Figure 1: Logged sources of information

There are two reasons why we capture these sources of information only. First, all above data are needed if we want to capture user interaction with its environment and with other users, as well as failure information. Second, due to security restrictions in the Symbian OS, we would have to certify our program with Symbian Foundation if we wanted to log information such as the cell identifier the phone is connected to, or the signal strength of the 3G/HSDPA.

Logging.

Since the program runs on limited resources, much care has been taken to have the smallest possible footprint on the system. The program is divided in two: a graphical user interface, to start/stop the service and to modify logging parameters, and the logger application itself.

The graphical user interface is simple, and permits the user to stop the logging service when he/she wants, modify the frequency of scan for every data source, and turn on or off the logging for every parameter. A screen capture is shown in Figure 2.

Figure 2: GUI for managing logger frequency

The logger is programmed using one thread only, by using the concept of Active Objects. Active Objects is a system mechanism that provides a comprehensive way to perform multiple tasks using system calls (e.g., reading of system parameters such as GPS information) within a single thread that acts as a scheduler. Hence, the resulting program uses low system resources and optimizes battery usage.

The thread is programmed to scan all parameters shown in Figure 1, and to store all information in a text file. The chosen data format is a simple human readable line-based log, that shows for each scanned parameter the time of logging, the type of logging, and the value of the log, as shown in Figure 3.

```
Starting scan
current parameters#GPS#1#WIFI#1#BATTERY#1#CALL#1#BT#1#
2009-10-2 13:50:39#GPS#+43.59407#+1.46388#+136.49504#+25.50000
2009-10-2 13:50:39#BATTERY#71#EPoweredByBattery
2009-10-2 13:50:39#POWER#BATTERY
2009-10-2 13:50:51#BT#berimbau#00:02:26:60:08:8b
2009-10-2 13:50:57#BT#euclide#00:01:14:45:51:1d
2009-10-2 13:50:58#BT#Nokia marco#00:02:24:40:04:4d
2009-10-2 13:51:23#BT#Mattmobile#00:02:21:1f:fc:c3
2009-10-2 13:51:32#BT##00:02:25:50:00:0c
2009-10-2 13:51:39#WIFI#Network 1#60#open#Infrastructure#\
        00:23:33:78:7f:60#laas-welcome
2009-10-2 13:51:39#WIFI#Network 2#91#open#Infrastructure#\
        00:25:45:b5:75:00#laas-welcome
```

Figure 3: Example of log

Gathering.

The program that runs on the mobile stores information locally. We are currently developing a networking part, that will opportunistically send logs to a secured server in the laboratory. Although sending information to a server seems a simple task, this requires additional work, due to security risks put on users, particularly for privacy reasons. We are investigating cryptographic mechanisms that would protect users' privacy, while still permitting analysis on gathered data.

Obviously, we cannot just anonymize every data trace with a fixed random identifier, because each trace contains many personal information (localization, users interactions) that could be used to "de-anonymize" a trace [6].

3. ANALYSIS AND USE OF COLLECTED DATA

Once the privacy-preserving database is filled with user data, it can be used for different tasks: analysis of data for social networks, resilience evaluation of mobility-aware algorithms, and privacy preserving analysis of the database.

Social Links Engineering.

In the vein of recent work on social links engineering [17] [7] [5], we plan to use this database, that contains more information than operators' mobile phones logs, to derive realistic mobility models. Current mobility models are mostly random-based and thus do not take into account the fact that devices are carried by humans.

In fact, the database will not only be useful for mobility modeling but, more generally, will serve as a basis for social links analysis, and particularly to answer the following questions:

- Does there exist *social locality*?, i.e., when social links exist between a group of users, do these links imply a more frequent co-locality in the physical world ? Such a result would have a strong impact on possible extensions of current social services (MySpace, Twitter, Facebook) to geo-localized systems. Intuitively, such locality principle should exist. However, measurements would confirm or infirm this hypothesis, and would allow to develop models that efficiently capture this potential locality.

- Can we model expected *interaction patterns* between users, so that we can build collaborative services that are based on stable users interactions ?

- Are there patterns of interactions between users and the environment that are more likely to appear than others ?

Replying to these questions seems for us a prerequisite to be able to design efficient and useful services for distributed systems of mobile users.

Resilience evaluation.

Collected data will also serve as an input both for our research on analytical evaluation of resiliency of algorithms [3] and for our reduced-size experimental platform [11]. For analytical evaluation, we will compute probabilistic laws for useful evaluation parameters, such as the expected time of users interaction, the expected time between failures, etc.

For experimental evaluation, we are interested in isolating *mobility patterns* that will then be used as an input to our experimental platforms, with the final goal of performing controlled and realistic experimentation by testing different algorithms on the same mobility patterns, a task that is impossible in a real environment.

Privacy evaluation.

The last use of the collected data we foresee in the actual evaluation its privacy-preserving features. From the very start of the project, we were struggled by privacy issues. Research results [6] [1] show that simple privacy-preserving strategies do not suffice to protect users, due to the huge amount of personal identifying data traces stored.

The more traces collected, the higher the threat on users' privacy. As technology evolves, eventually almost everyone will be able to dig into such information, and providing users with means to protect their data will become a priority. Performing attacks on such a database will permit us to detect wether our strategies for pseudonymation and privacy-guarantee is robust.

4. CONCLUSION

We presented the design and implementation of a mobile phone powered data collection platform. Our platform collects data that, we believe, will ease the understanding and modeling of several phenomenons related to human activity: social links, mobility, etc.

We still have to define the economical nature of data collection. Users hardly want to have their privacy exposed without a counter part. Such reward could consist of several parts: (1) an economic part, by providing a free terminal, (2) a service-oriented part, by providing services users are not used to, such as free videoconference service, and, as we hope, (3) an altruist academical part, providing users the satisfaction to participate to a research program. It is worth noticing that the chosen "remuneration scheme" will imply a bias on collected data that will have to be studied...

5. ACKNOWLEDGEMENTS

The authors would like to thank Hamdi Ayed for its implication in the development of the first version of the software.

6. REFERENCES

[1] L. Backstrom, C. Dwork, and J. Kleinberg. Wherefore art thou r3579x?: anonymized social networks, hidden patterns, and structural steganography. In *WWW '07: Proceedings of the 16th international conference on World Wide Web*, pages 181–190, New York, NY, USA, 2007. ACM.

[2] C. Boldrini, M. Conti, and A. Passarella. *Social-based autonomic routing in opportunistic networks*, pages 1–37. Springer, 2009.

[3] L. Courtès, O. Hamouda, M. Kaâniche, M.-O. Killijian, and D. Powell. Dependability evaluation of cooperative backup strategies for mobile devices. In *Proceedings of the IEEE International Symposium on Pacific Rim Dependable Computing*, Melbourne, VA, Australia, December 2007.

[4] N. Eagle and A. S. Pentland. CRAWDAD data set mit/reality (v. 2005-07-01). Downloaded from http://crawdad.cs.dartmouth.edu/mit/reality, July 2005.

[5] N. Eagle, A. S. Pentland, and D. Lazer. Inferring friendship network structure by using mobile phone data. *Proceedings of the National Academy of Sciences*, 106(36):15274–15278, 2009.

[6] S. Gambs, M.-O. Killijian, and M. N. del Prado. Gepeto: a geo-privacy enhancing toolkit. In *Proc. of the Int. Workshop on Advances in Mobile Computing and Applications: Security, Privacy and Trust, 24th IEEE AINA conference, Perth, Australia, April 2010*. IEEE, 2010.

[7] M. C. Gonzalez, C. A. Hidalgo, and A.-L. Barabasi. Understanding individual human mobility patterns. *Nature*, 453(7196):779–782, June 2008.

[8] D. Hay and P. Giaccone. Optimal Routing and Scheduling for Deterministic Delay Tolerant Networks. In *WONS'09: 6th. Int. Conf. on Wireless On-demand Network Systems and Services*, pages 25–32. IEEE, 2009.

[9] M. J. Keeling and L. Danon. Mathematical modelling of infectious diseases. *British Medical Bulletin*, 92(1):33–42, DEC 2009.

[10] M.-O. Killijian, D. Powell, M. Banâtre, P. Couderc, and Y. Roudier. Collaborative backup for dependable mobile applications. In *Proceedings of 2 nd International Workshop on Middleware for Pervasive and Ad-Hoc Computing (Middleware 2004)*, pages 146–149, Toronto, Ontario, Canada, October 2004. ACM Press.

[11] M.-O. Killijian and M. Roy. A platform for

experimenting with mobile algorithms in a laboratory. In *ACM Conference on Principles of Distributed Computing (PODC'09)*, pages 316–317. ACM, 2009.

[12] S. Merler and M. Ajelli. The role of population heterogeneity and human mobility in the spread of pandemic influenza. *Proceedings of the Royal Society B: Biological Sciences*, 277(1681):557–565, FEB 22 2010.

[13] M. Musolesi and C. Mascolo. Mobility models for systems evaluation. In *State of the Art on Middleware for Network Eccentric and Mobile Applications (MINEMA). Springer, February 2009.*

[14] M. Piorkowski, N. Sarafijanovic-Djukic, and M. Grossglauser. CRAWDAD data set epfl/mobility (v. 2009-02-24). Downloaded from http://crawdad.cs.dartmouth.edu/epfl/mobility, Feb. 2009.

[15] M. Raento, A. Oulasvirta, R. Petit, and H. Toivonen. Contextphone: A prototyping platform for context-aware mobile applications. *IEEE Pervasive Computing*, 4(2):51–59, 2005.

[16] M. Roy, F. Bonnet, L. Querzoni, S. Bonomi, M.-O. Killijian, and D. Powell. Geo-registers: An abstraction for spatial-based distributed computing. In *Int. Conf. On Principle of Distributed Systems (OPODIS'08)*, volume 5401 of *Lecture Notes in Computer Science*, pages 534–537. Springer, 2008.

[17] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi. Limits of Predictability in Human Mobility. *Science*, 327(5968):1018–1021, 2010.