MobileSens: A Framework of Behavior Logger on Andriod Mobile Device

Rui Guo[†‡], Tingshao Zhu[*†], Yu Wang[†‡], Xinguo Xu[‡]
[†]*Graduate University of Chinese Academy of Sciences, Beijing 100190, China*
[‡]*The 6th Research Institute of China Electronics Corporation, Beijing 100083, China*

Abstract

To provide a better service, it is important to understand user's behavior on mobile device. In this paper, we propose a framework for logging, MobileSens, to record users' operations on Andriod mobile device, including smart phone and pad. The operations include turning on screen, sending messages, and browsing the web, etc. This system uploads the data to server via GPRS(General Packet Radio Service) for further analysis. MobileSens monitors system events, and identifies any event that related to the user behavior. Recording the usage information could help us understand the user's intention and mental status, thus refine the user interface, then make these mobile devices serve people better.

Keywords: behavior logger; pad; smartphone; mobile device.

1 Introduction

According to the latest report, there are more than 50 million mobile phone users around the world, about 72.46% of the whole population. The ratio of mobile phone users in developed countries is about 97% in 2007, 45% in developing countries (still growing). With the development of technology, mobile devices become more and more user-friendly and intelligent. For example, the smartphone not only provides the main functions of voice and text communication, but also provides network, multimedia, entertainment and other services. It has become a main communication tool and a part of digital life.

To provide a better service, it is important to understand user's behavior on mobile device. In this paper, we propose a framework for logging, MobileSens, to record users' operations on Andriod mobile device, including smart phone and pad.

To obtain the user's behavior and context while using mobile device, such as the duration on visiting web page, the frequency of changing the background, we designed the system MobileSens to record the smartphone usage information. The system consists of two parts. The first part is on client side, which is a program running background, capturing the usage information. The second part is on server side, to keep the usage information for any further analysis.

The rest of this paper is organized as follows. In Section 2, we introduce some related works, others have made before. Section 3 describes the architecture of MobileSens. We present several implementation issues in section 4 and discuss the conclusion and future works in section 5.

2 Related Work

To investigate the user's behavior [5] [11], especially for large scale user studies [6], several logging tools have been developed.

Robert *et al* [8] implemented WebEyeMapper and WebLogger to record both the eye movements on web pages, and the actual page sequence. It can capture the user's action on IE browser, and save the viewed web content as well. Pirolli *et al* [7] proposed a user-tracing architecture which can be used to record the user's behavior on the web. Jibo [4] implemented a web browser, SurfLogger, which is based on Python, but it might take much time to start up. João and Jorge [10] have developed another WebLogger which is attached to event handler, thus to capture these major browser-events, but it may miss some events when the user performs on the browser directly. These web trace logging tools either capture system events or host the browser, but on mobile device, web browser is only one application that the user may operate. We need to record much more operations on the device, besides browsing behavior [2].

SystemSens [1] is a smartphone logging system. It can be deployed on Android smartphone, sending the colleted

* Corresponding Author: Tingshao Zhu, *tszhu@gucas.ac.cn.*

978-1-4577-0208-2/11/$26.00 ©2011 IEEE

data to the server through the network. Its target is to measure the usage of various resources in user's smartphone, such as CPU, memory, network, etc.

LiveLab [9] is designed to record real-world smartphone usage with a reprogrammable built-in logger designed for long-term user studies. It runs on the iPhone. It can capture web history, currently running processes, available wifi access points and GPS Location. The target of LiveLab is to analyze the features of application usage, network conditions and net usage. Such context dependency provides key insights into the optimization of the mobile and network system.

MyExperience [3] is a BSD-licensed open source mobile data collection tool developed for Windows Mobile devices (including PDAs and mobile phones). MyExperience combines sensing and self-report to collect both quantitative and qualitative data on human behaviors, attitudes and activities in the field. Using a mobile phone's wireless connection to the internet, researchers have the ability to access MyExperience log data, allowing for ongoing analysis of data and early detection of subject compliance or technology issues.

The table 1 shows the main parameters of the three data collection tools.

Table 1. List parameters of three data collection tools

System	OS	Content	Data Collection	Intervention
MyExperience	Windows mobile	User experience, Usage	Mobile deveice or Web server	Yes
SystemSens	Android	Usage	Web server	No
LiveLab	iPhone iOS	Wireless network, usage	Web server	No

In short, most current logging systems aim to collect system usage information, and they are mainly designed to optimize the system performance. While in our research, we intend to investigate how the user's behavior on these mobile device, and provide better service to help people. Therefore, our logging system focuses on user behavior, rather than system usage information.

3 MobileSens Architecture

Our logging system, MobileSens, consists of two parts: client side and server side. The client side runs on the mobile device(i.e., smartphone and pad), to log the user's behavior. While the server side is on web server, which receives the data from the client and saves it into database. Fig. 1 depicts the systems structure.

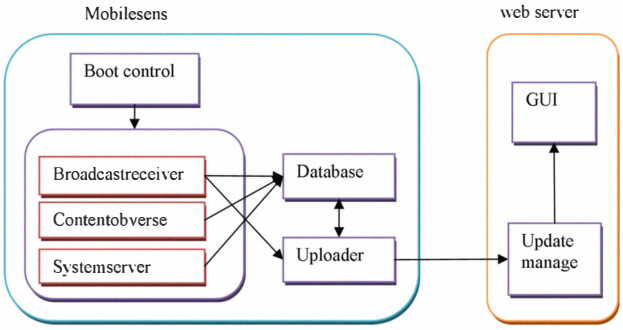

Figure 1. The Architecture of MobileSens

Client side is running on the Android. It is a service application and has no influence on normal operation. In addition, it only takes very low system resource. We capture the usage data though the API provider by the android. The client requires the smartphone to access the internet and uploads data to web server though the internet. Our aim is to facilitate the communication and to reduce the cost of storing data.

The client 's life cycle is determined by android, with the slight differences among different versions of android. In order to capture more information the client part need to be run on the android2.2 or later. The client part sends data through internet to web server in the way of "POST". The data is sent when the client received the signal that the screen is shut down. When the data is uploaded successfully, it will be removed from local database.

The server side is mainly for logging data storage. We install MySQL database and Apache on server. When the server receives JSON objects form client, it decodes the data through PHP language and writes them into the local database(Sqlite), and then upload to the server. As Fig. 4 shows, we are able to access those data and conduct more analysis.

4 Implementation Issues

MobileSens is implement on Java, and runs on Android 2.2 and later. There are several implementation issues discussed as follows.

4.1 MobileSens Client

The client consists of four modules: boot control, sensor, database and update module.

Boot control module consists of two parts. The first part receives the system starting success signal to start MobileSens. The other part periodically captures the alarm signal from the MobileSens to drive MobileSens for logging.

Sensor module acts as a virtual sensor, and it obtains the system information from various API providers of android OS.

Database module is responsible for the data storage. It writes the information to the data tables on local Sqlite3 database. MobileSens captures various types of information. There are significant differences between each type of information. In order to manage the information effectively, we encode data into a JSON object and form a unified data format. The data table is designed like this(id integer primary key "+" auto-increment, time text not null, "+" type text not null, msg text not null). The time is record time. The type is information type. The msg is a JSON object of the information body.

Mobilesens is a system server application. It needs to be triggered. When it receives the system boot-completed signal, the boot control module creates a server dalvik, and registers many standard broadcast receivers, content observers, system servers. Like standard service application, MobileSens runs on background. When it receive a broadcast which is ACTION_BATTERY_CHANGED, it will upload usage data to the web server. In the course of MobileSens operation, the boot control module receives the alarm signal sent by MobileSens-self to drive sensors to capture the system logging events.

4.2 Logging

MobileSens provides four ways for logging. The first one is to receive the OS broadcast, such as new outgoing call, power connects or disconnects, screen open or close, etc. The second is to read system logger periodically, such as Network traffic, activity application logger, system server logger. In MobileSens, the interval is two minutes. The third is to register a content observer. The content provided by Android can share data between difference applications. In MobileSens, Sensor module actually monitors the change information of these data by content observer. The fourth is to register the system server. There are many system services, such as power manager, alarm manage, location provider. For example, we get GPS information through registering a location listener to listen to location providers.

Tab. 2 summarizes the logging capability of MobileSens. The table also lists the logging method and content for each type.

We chose JSON to transfer data between client and web server. JSON gives us the flexibility to add new types of sensors and data records without changing database schema. In addition, it is more efficient to handle JSON objects than XML on Android. Fig. 2 shows an example of the JSON object. Each JSON object has four pairs of key-value: information record time, timestamp, information type, and IMEI of smartphone which is used to identify the user. The "msg" is the information body.

```
{
  'date':'2011-2-22 0:42:56',
  'timestamp':'1298364176416',
  'logtype': 'powerLog',
  'imei': '355060041008892',
  'msg': {'status':'power off'}
}
```

Figure 2. The JSON object for transferring data

4.3 Uploading

The most energy and resource consuming task of the MobileSens is uploading the records to the server because of high energy consumption associates with network transmission. The solution in SystemSens [1] is to upload only when the phone is being charged. The benefit of this upload mechanism is to offer application enough opportunity to upload all the data, because most smartphone users charge their phones overnight and have zero interaction with their phones. As mentioned in the paper, this scheme would fail if a user charges his phone while it is turned off, or at a location with failed network connect, or for users with multiple batteries which charge them outside the phone or etc.

In addition to upoading while being-charged, MobileSens uploads data when screen is off as well. When screen is off, the system consumes minimum source to run and has zero interaction information.

4.4 Deployment

After Implementation, we run MobileSens on Google Android's SDK (android2.2), Motorola MT-016 (an-

Table 2. List of the type, name, content and method of logging in MobileSens

type	name	content	method
"activityLog"	Activity application logger	Record the careate, start, resume, stop, exit of activity application.	Read system logger
"apkLog"	Application package logger	A package has been added, change, and remove.	Broadcastreceiver
"callLog"	Calling logger	Record the state,number,contact,direction of calling	Broadcastreceiver
"camerabuttonLog"	Carmera button logger	Whether a camera button has press.	Broadcastreceiver
"configuationLog"	Configuation logger	Record the configuation changed information.such as font, screen size, keyboard type.	Broadcastreceiver
"contactLog"	Contacts loggger	Record the added, changed, delete of contacts.	Contentobserver
"datechangedLog"	Date changed logger	Date and time changed information	Broadcastreceiver
"gpsLog"	GPS logger	The information of the user locale, altitude, latitude, longitude and bearing.	Systemserver
"headssetLog"	Headset logger	Record the Headset plugged in or unplugged.	Broadcastreceiver
"localechangedLog"	Locale changed logger	Recored the locale changed.include language and State	Broadcastreceiver
"netLog"	Data transfer logger	Record the amount of data each appplication send and receive	Read system logger
"powerconnectedLog"	Charge logger	The information of the power connected and disconnected	Broadcastreceiver
"powerLog"	Power logger	Record information of the smartphone power on and power off	Broadcastreceiver
"screenLog"	Screen logger	Record the state of the screen.	Broadcastreceiver
"serviceLog"	Service application logger	Record create, start destory of the service application.	Read system logger
"smsLog"	SMS logger	Record the state, content, contacts of SMS.	Broadcastreceiver
"wallpaperchangedLog"	Wallpaper logger	The information of the wallpaper changed	Broadcastreceiver
"wifiLog"	Wifi logger	The information of wifi access point.	Broadcastreceiver
"urlLog"	Access internet logger	The information of user visit internet. Such as url, title, times, bookmark.	Contentobserver

droid2.1) and Samsung GT-I5508 (android2.2). Fig. 3 shows MobileSens running on SAMSUNG smartphone. On Android 2.1, we can't obtain currently running active program log, service programs log, outgoing call information. That is because the broadcast information and system services of these three items are not available on Android2.1 and earlier. In the above tests, our program can run smoothly and continuously in the simulator and the actual smartphone. After connecting to the Internet, our program is able to upload recorded information to the web server properly. We can access the data from webpage on server. Fig. 4 presents the web pages that display these logging data.

While deploying MobileSens, There are several concerns that we should pay attention to.

Workload Mobilesens is able to get 19 kinds of user's information, which is a large amount of data. The data size is determined by the interaction between users and device. The Large-scale data transmission may decrease the device's performance and working life, we plan to do more computation on the device, and reduce the data transmission.

Privacy Since the system is designed to get the user's records as much as possible, a lot of obtained information has an invasion of user's privacy, such as incoming and outgoing calls, GPS information, text messages and so on. In the future, it enables the user to

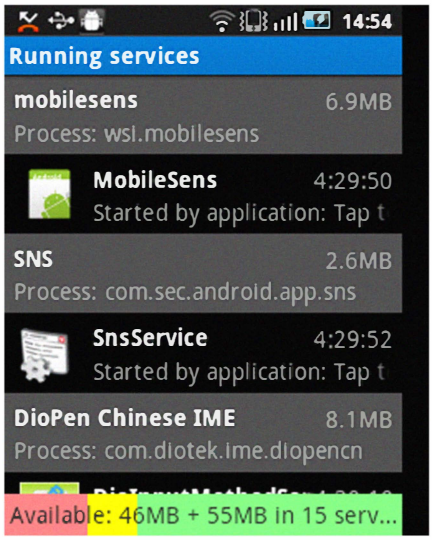

Figure 3. MobielSens running on Samsung GT-I5508

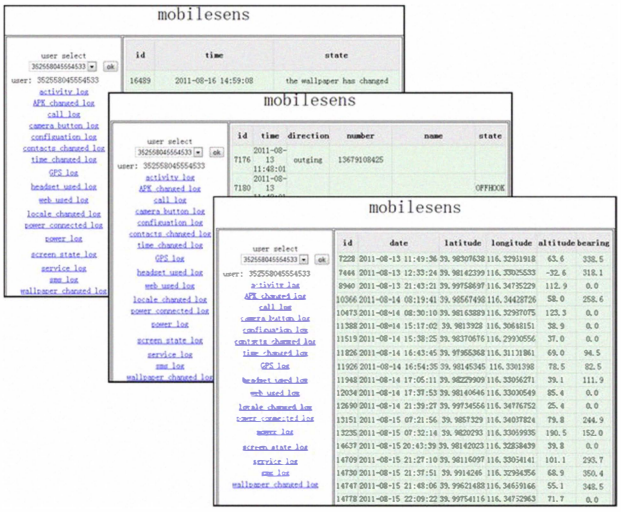

Figure 4. The Logging data of GPS, calling, wallpaper on the server

decide which information allows for logging. At the same time we will process the information to protect user's privacy.

Security In MobileSens, we use POST transmission, which may has potential security problems. In the new version, we plan to try more secure method for data transmission.

5 Conclusion

Recording the usage data of mobile device could be used further analyze user's behavior, to understand the user's habits, and optimize the whole system. This paper describes a recording system which can obtain the usage information in real time. We describe the design and implementation of Mobilesens in detail, and summarize the system's actual operating results and existing problems.

In the future work, further optimization of the system is required. We will allot small-scale data processing in mobile to reduce the data transmission. Additionally, we plan to add feedback in the transmission of information to ensure information integrity and protect user's privacy. According to the actual user's use condition, we can decide which information is needed to be captured. So we could obtain meaningful data as much as possible which can be used in other related researches.

6 Acknowledgment

The authors gratefully acknowledges the generous support from NSFC(61070115), Research-Education Joint Project(110700EA02) and 100-Talent Project(110700M202) from Chinese Academy of Sciences.

References

[1] Hossein Falaki, Ratul Mahajan, and Deborah Estrin. Systemsens: a tool for monitoring usage in smartphone research deployments. In *Proceedings of the sixth international workshop on MobiArch*, MobiArch '11, pages 25–30, New York, NY, USA, 2011. ACM.

[2] Hossein Falaki, Ratul Mahajan, Srikanth Kandula, Dimitrios Lymberopoulos, Ramesh Govindan, and Deborah Estrin. Diversity in smartphone usage. In Sujata Banerjee, Srinivasan Keshav, and Alec Wolman, editors, *MobiSys*, pages 179–194. ACM, 2010.

[3] Jon Froehlich, Mike Y. Chen, Sunny Consolvo, Beverly Harrison, and James A. Landay. Myexperience: a system for in situ tracing and capturing of user feedback on mobile phones. In Edward W. Knightly, Gaetano Borriello, and Ramn Cceres, editors, *MobiSys*, pages 57–70. ACM, 2007.

[4] Jibo He. Surflogger: A logging browser and data processing method in web-based studies. In *Proceedings of Society of Computer in Psychology, Chicago*, 2008.

[5] Amy K. Karlson, Brian Meyers, Andy Jacobs, Paul Johns, and Shaun K. Kane. Working overtime: Patterns of smartphone and pc usage in the day of an information worker. In Hideyuki Tokuda, Michael Beigl, Adrian Friday, A. J. Bernheim Brush, and Yoshito Tobe, editors, *Pervasive*, volume 5538 of *Lecture Notes in Computer Science*, pages 398–405. Springer, 2009.

[6] Earl Oliver. The challenges in large-scale smartphone user studies. In *Proceedings of the 2nd ACM International Workshop on Hot Topics in Planet-scale Measurement*, HotPlanet '10, pages 5:1–5:5, New York, NY, USA, 2010. ACM.

[7] Peter Pirolli, Wai-Tat Fu, Robert Reeder, and Stuart K. Card. A user-tracing architecture for modeling interaction with the world wide web. In *Proceedings of the Working Conference on Advanced Visual Interfaces*, AVI '02, pages 75–83, New York, NY, USA, 2002. ACM.

[8] Robert W. Reeder, Peter Pirolli, and Stuart K. Card. Webeyemapper and weblogger: tools for analyzing eye tracking data collected in web-use studies. In *CHI '01: CHI '01 extended abstracts on Human factors in computing systems*, pages 19–20, New York, NY, USA, 2001. ACM.

[9] Clayton Shepard, Ahmad Rahmati, Chad Tossell, Lin Zhong, and Philip T. Kortum. Livelab: measuring wireless networks and smartphone users in the field. *SIGMETRICS Performance Evaluation Review*, 38(3):15–20, 2010.

[10] Jo˜ao Silva and Jorge Bernardino. Simplifying the clickstream retrieval using weblogger tool. In *ICWI*, pages 444–451. IADIS, 2004.

[11] TIMO SMURA. Access alternatives to mobile services and content: Analysis of handset-based smartphone usage data. In *ITS 17th Biennial Conference*, 2008.