Mobile network measurements using Android

Fernando Molina Albero[1], Andrej Štern[2], Andrej Kos[2]

[1] University Miguel Hernández, The Polytechnic School of Elche (EPSE), Spain
[2] University of Ljubljana, Faculty of Electrical Engineering, Laboratory for Telecommunications (LTFE)
E-mail: fernando.molina@alu.umh.es

Abstract

Introducing a mobile telephony standard requires the cooperation of many engineers. Once implemented the networks require many improvements on the long-term scale. Lots of research is performed at the universities and similar research institutions with limited access to expensive measurement equipment. Introducing the new generations of mobile devices, e.g. smart phones, enables the development of applications with functions which can be compared to sophisticated professional measurement systems.

This paper shows the possibilities of optimizing the process of measurement by using a widely spread Android devices. By using the open source tools the measurement applications can be offered to a vast community providing huge datasets and lowering the overall measurement costs.

1 Introduction

The last installed mobile technology in Slovenia is 3.9G LTE, currently in the phases of coverage improvements and other network optimisations. It is important to constantly monitor the system's coverage and available capacity in order to reach a high level QoS. In the past, network operators used solely drive tests that require large Operation Expenditure (OPEX), while the collected measurements can only provide limited snapshots of the entire network. This drive tests were based on tracing and collecting information from a measurement car where only selected roads could be scanned with limited geographical scope. Usually, the indoor situations were simulated through the signal penetration from the outdoor.

From Release 9 onwards the 3GPP and the Next Generation Mobile Networks (NGMN) alliance have set standards and recommendations for performing the drive tests, grouped under the common name MDT (Minimization of Drive Tests) [1]. MDT exploits the routinely measured signal quality information at all mobile phones in order to make decisions about connectivity optimization. It is possible to take measurements while in active state, thus providing more parameters, and even in idle mode, where results are logged to internal memory and sent postponed. These MDT functionalities are still being developed in R12, so the time is needed to achieve global penetration by implementing new releases to new phones.

With the introduction of smart phones the situation has changed. Every phone can carry a tiny application that is scanning its surroundings in timely manner. There are lots of downloadable applications available for the Android operating system, which gains popularity among mobile users. The figure 1 presents the high growth of worldwide Android sales [2, 3]. In parallel, the increasing number of users represents potential application developers and testers. By introducing applications to Google Play, the Android community can evaluate software easily and even provide valuable information for upgrading it to the advanced level.

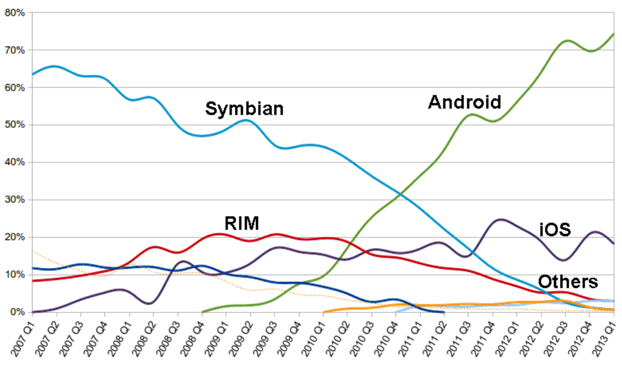


Figure 1: Worldwide Smartphones Sales %

2 Motivation for work

Due to my staying as exchange student at the Faculty of electrical engineering, Ljubljana, the decision was made that I prolong the last study year for finishing my final project as thesis. In accordance to my home faculty at Miguel Hernandez University, Spain, we decided to perform a research and development in the fields of mobile telecommunications. The topics of merging Android programming with testing the mobile networks were covered by LTFE team.

Since I am an Android user and have programming skills we have decided to devote my final parts of study to analyze the existing open source network monitors and establish a new mobile application with improved performances. Our goal is to establish the framework for accurate measurements of radio interfaces in a form of over-the-top services which don't rely on operator's cooperation or standardization. This paper outlines the progress of student's work, which is still under development.

3 Android architecture

The Android OS can be represented as a software stack of several layers, where each layer is a group of cooperating program components. Together it includes operating system, middleware and system applications.

Android OS is actually limiting the types of different measurements we can take from the environment. An access to the hardware's radio and modem is provided by the Radio Interface Layer (RIL). The Android Open Source Project [4] provides a RIL between Android telephony services API "*android.telephony*" and the radio hardware. The RIL consists of two main components: a RIL Daemon and a Vendor RIL. The RIL Daemon talks to the telephony services and dispatches solicited commands to the Vendor RIL. The Vendor RIL is specific to a particular radio implementation and dispatches unsolicited commands up to the RIL Daemon, as shown on the figure below.

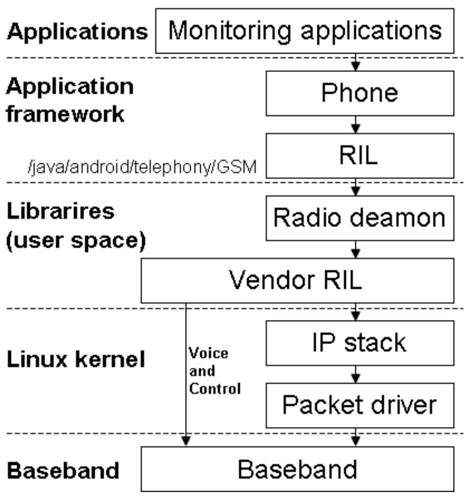


Figure 2: Radio Interface Layer

There are several versions of Android OS that support different API levels (Table 1). The older phones had typically installed Android version 2.3.3 (Gingerbread) which supports the limited API level 10. The telephony classes inside "*/java/android/telephony*" for the API 10 are presented in Table 2 [4]. The latest versions of Android OS are known as 4.2.2 and 4.3 (Jelly Bean) using API 17 and 18 (Table 3) [4]. They introduce the whole new set of measurements in addition to older APIs including LTE measurements and special cell identities.

Table 1: Android versions and APIs

Android version	Codename	API
2.3.x	Gingerbread	10
3.2	Honeycomb	13
4.0.x	Ice Cream Sandwich	15
4.1.x	Jelly Bean	16
4.2.x		17
4.3		18

Table 2: API 10 Telephony Classes

CellLocation	Abstract class that represents the location of the device.
NeighboringCellInfo	Represents the neighboring cell information, including Received Signal Strength and Cell ID location.
PhoneNumberUtils	Various utilities for dealing with phone number strings.
PhoneStateListener	A listener class for monitoring changes in specific telephony states on the device, including service state, signal strength.
ServiceState	Contains phone state and service related information.
SignalStrength	Contains phone signal strength related information.
TelephonyManager	Provides access to information about the telephony services on the device.

Table 3: API 17 Telephony Classes (in addition to API 10)

CellIdentityCdma	CellIdentity is to represent a unique CDMA cell
CellIdentityGsm	CellIdentity to represent a unique GSM cell
CellIdentityLte	CellIdentity is to represent a unique LTE cell
CellIdentityWcdma	CellIdentity to represent a unique UMTS cell
CellInfo	Immutable cell information from a point in time.
CellSignalStrength	Abstract base class for cell phone signal strength related information.

According to Android develop web page [5] the widely implemented versions of Android are still 2.3.3 and 4.1.x. Officially the upgrades of Gingerbread to Jelly Bean are not supported unless users decide to void the warranty and install proprietary cooked ROMs (e.g. CyanogenMod). So the majority of old phones can measure only parameters supported by telephony classes in table 2. Inside the 4.1 to 4.3 versions the upgrades are easy because phone manufacturers decide to upgrade it sooner or later.

4 Observed LTE parameters

The LTE measurements are supported only in latest releases of APIs. From the radio perspective (e.g. OFDM, signal structure, bandwidths) LTE is a new technology, so new types of parameters can be acquired. These are mostly related to signal power, quality and special counters, as presented below.

RSRP (Reference Signal Received Power) is defined as the linear average over the power contributions in Watts of the resource elements that carry cell-specific reference signals within the considered measurement frequency bandwidth. We can also consider RSRP as the "absolute strength" of your current connection.

RSSI (Received Signal Strength Indicator) is the linear average of the total received power in Watts. This information was shown on all devices as the dBm/ASU "signal status" under the Phone Info menu.

SNR (Signal to Noise Ratio) is the ratio of signal strength to noise. The higher your SNR, the more throughput (DL/UL speed) your connection will have.

CQI (Channel Quality Indicator) is a measure of the quality for the current channel in the camping cell. CQI is derived from the SNR and the SINR.

RSRQ (Reference Signal Received Quality) is the overall quality of your signal in general. RSRQ ranges from -3db to -19.5db with a number closer to -3db being better.

SINR (Signal Interference Noise Ratio) shows how good your connection is (the stability of it and how close it is to handing off to 3G).

Cell handover is used to measure how many jumps there are between neighbouring cells to discover certain instabilities of LTE network.

5 Our approach

For the development of the application we are using an Android device Samsung Galaxy S4 Mini (GT-I9195) [6]. It is compatible to GPRS, EDGE, UMTS (W-CDMA), HSxPA and LTE standardization. The programming computer uses the Linux Ubuntu 13.04 OS, since the Eclipse [7] environment here is more stable than on Windows OS (own experiences). For the programming we use Google's SDK and ADT [8]. These Android packages provide the possibility to develop applications using Telephony API, debugging interface and convenient environment for testing the compatibility to the different terminals with Android. All used software is covered by open source licences so there are no additional costs for the developers.

Developing in Android is quite easy with knowledge about HyperText Markup Language (HTML) because Android uses a very similar XML for layouts and JavaScript for the engine of the application. Our application architecture is shown in the figure below.

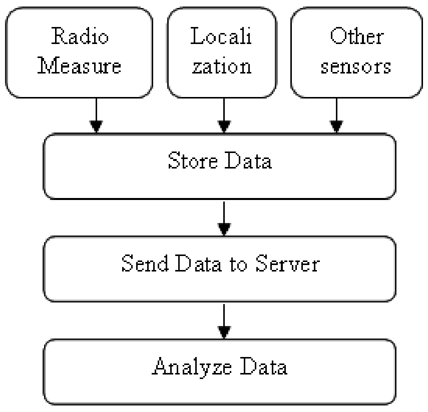
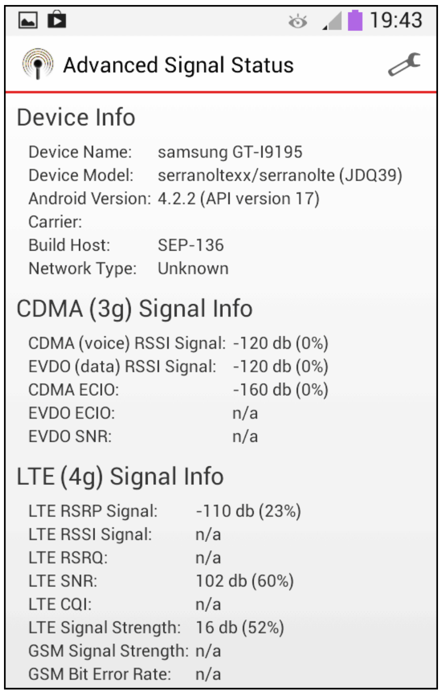


Figure 3: Our own application architecture

The initial programming for **radio measure** was based on the open source application "Advanced Signal Status" [9], available on Google Play and GitHub. This application already shows some of the radio measurements of different radio access networks like presented in Figure 4. In order to optimize measuring procedures additional parameters were added.


19:43

Advanced Signal Status

Device Info

Device Name:	samsung GT-I9195
Device Model:	serranoltexx/serranolte (JDQ39)
Android Version:	4.2.2 (API version 17)
Carrier:	
Build Host:	SEP-136
Network Type:	Unknown

CDMA (3g) Signal Info

CDMA (voice) RSSI Signal:	-120 db (0%)
EVDO (data) RSSI Signal:	-120 db (0%)
CDMA ECIO:	-160 db (0%)
EVDO ECIO:	n/a
EVDO SNR:	n/a

LTE (4g) Signal Info

LTE RSRP Signal:	-110 db (23%)
LTE RSSI Signal:	n/a
LTE RSRQ:	n/a
LTE SNR:	102 db (60%)
LTE CQI:	n/a
LTE Signal Strength:	16 db (52%)
GSM Signal Strength:	n/a
GSM Bit Error Rate:	n/a

Figure 4: The initial Advanced Signal Status app.

Collecting the measurements requires the **localization**. The best technology for positioning is surely derived from GPS/Glonass receiver inside the phone. Receiving satellite signals is only possible outdoors or near the open surfaces (e.g. windows). It can seriously impact the phone's autonomy since the power consumption of GPS module is high compared to the LTE reception. There is another solution that comes from Google and is limited by accuracy. By using Google's Location API it is possible to locate a mobile device using neighbouring cell towers, Wi-Fi access points or even GPS outdoors. It always works using best-effort principle so the externally required accuracy can not be guaranteed.

Optionally, the data from **other sensors** can be collected. The phone already includes several sensors, e.g. accelerometer, gyro, proximity, compass, light and temperature sensors etc. This information might come handy when a decision, based on measurements, has to be taken but the primary means don't provide enough reliability.

The **storage** is needed when we want to keep the information inside the terminal without uploading it to the server. Some cases, when immediate uploading is not preferred, are:

- the data consumption over mobile network: sometimes it is better to wait for the available Wi-Fi signal when real-time measurements and reporting is not required,
- the energy consumption: transmitting has always high demands on battery management so appropriate transmission models must be respected.

The data storage needs special user permissions. The Android OS is prepared for these things: when the user installs the application, it only has to accept the permissions for the application. These permissions are also required for using the localization and telephony API. The data is saved in a file within the phone's SD card in special formats that occupy as little space as possible (e.g. special databases).

The **server** collects the received data from different terminals and possible other services for further analysis. In our case the communication to the server uses efficient FTP protocol.

The data from the server is **processed** by a script that transfers received data to common databases (e.g. My SQL) and presentation tools. For the presentation we are using a commercial software Tableau [10] which is capable to process large amounts of data and provide intuitive interfaces in form of graphs and statistical calculations. So the presentation of measured data is easily done in a timely manner.

6 Conclusion

The necessity for automated network measurements is surely present in all generations of mobile networks. The standardization is evolving towards measurement tools based on user equipment in a form of smartphones. A new approach using smartphone's APIs shows the possibilities where the whole community can take part in data collection and reporting back to data servers.

The analytical approach to this topic showed that there are numerous of Android OS versions which differ in means of supporting telephony classes. Surely the latest releases include additional information about newest LTE technologies that are available only from the Android version 4.2 onwards presenting API 17.

This paper presented the intermediate stage of student's thesis in a form of final project that will be finished only after the presentation on ERK 2013. At the time of writing the planned application was still not mature enough to be explained in detail. But during the conference authors plan to make a demonstration.

Due to a fact that this application will remain open source, there will be lots of possibilities for other researchers and developers to improve and develop it further in order to move the state of technology another step higher.

Acknowledgements

The work was in part supported by the Slovenian Research Agency (ARRS, applied project »Quality of service and quality of experience measurement and control system in multimedia communications environments«, Grant nr. L2-4289).

Literature

[1] W. A. Hapsari et all. "Minimization of Drive Tests Solution in 3GPP," Communications Magazine IEEE, pp. 28–36, junij 2012.

[2] Figure 1 Reference. [Online]¸ [Cited: August 25, 2013] http://commons.wikimedia.org/wiki/File:World_Wide_Smartphone_Sales_Share.png

[3] Meulen, Rob van der. Gartner. [Online] [Cited: June 24, 2013] http://www.gartner.com/newsroom/id/2335616.

[4] Android for Developers [Online] [Cited: July 14,2013] http://developer.android.com/reference/android/telephony/TelephonyManager.html

[5] Android for Developers [Online] [Cited: July 14,2013] http://developer.android.com/about/dashboards/

[6] Samsung Company [Online] [Cited: July 05, 2013] http://www.samsung.com/uk/consumer/mobile-devices/smartphones/android/GT-I9195ZKABTU

[7] Eclipse Program, for programing. [Online] [Cited: July 05, 2013] http://www.eclipse.org/

[8] Android SDK web page. [Online] [Cited: July 05, 2013] http://developer.android.com/sdk/index.html

[9] Wes Lanning, in GitHub [Online] [Cited: July 13, 2013] https://github.com/yareally/SignalInfo

[10] Tableau Software [Online] [Cited: July 25, 2013] http://www.tableausoftware.com/