An Automatic and On-demand MNO Selection Mechanism

Christos Tsiaras, Samuel Liniger, Burkhard Stiller
University of Zürich, Department of Informatics (IFI), Communication Systems Group (CSG)
Binzmühlestrasse 14, CH–8050 Zürich, Switzerland
tsiaras@ifi.uzh.ch, samuel.liniger@uzh.ch, stiller@ifi.uzh.ch

Abstract—**A manual selection of the Mobile Network Operator (MNO) to be used on a mobile device is possible through the respective user interface. Furthermore, mobile devices can be adjusted to select automatically the MNO based on the strongest signal strength, among the list of those MNOs the Subscriber Identity Module (SIM) card is allowed to be registered with. However, so far in modern mobile operating systems, such as Android and iOS, there is no available method in the public developer's Application Programming Interface (API), which allows for an automatic and on-demand selection of the MNO by third-party applications. Recently, various research approaches assume the existence of an automatic and on-demand MNO selection mechanism to achieve different goals, such as breaking the termination rates monopoly (AbaCUS) or minimizing the non-ionizing radiation of mobile/wearable devices. The interest of such a mechanism has been raised three years ago by the Android developers community. Thus, this work here presents an automatic and on-demand MNO selection mechanism, that has been designed and implemented on the Android platform. For evaluation purposes the energy and end-to-end (e2e) time consumption while switching among MNOs using this mechanism is evaluated and as an applied example the data consumption of AbaCUS signaling messages is measured.**

Index Terms — **Android, mobile operators, selection mechanism, energy efficiency, data consumption**

I. Introduction

The development and the evaluation of an automatic and on-demand Mobile Network Operator (MNO) selection mechanism for the Android platform, which is presented in this work, is motivated due to the following 3 reasons: (1) The number of available MNOs in a certain location, (2) potential benefits of such a mechanism for the three main mobile communication stakeholders (MNOs, end-users, and regulation authorities), and (3) the existing number as well as the future estimation of devices in the mobile phones market that can support such a mechanism.

In 2013, 191 MNOs, which are active in 61 countries across Europe [22], result in an average of three available MNOs per country. Additionally, in mobile communications there is no physical barrier (*e.g.*, wires) that might force an end-user to stay connected with a specific MNO. Thus, MNO subscribers, due to multiple available MNOs in a location and commonly used medium in mobile communications, can hop automatically between different MNOs according to their preference.

From the MNO's perspective such a hopping attitude is driven by the fact that MNOs can benefit by offering on-demand premium services to any subscriber of any competitive MNO*, e.g.*, high and/or guaranteed sound quality or guaranteed access to the network in case of network congestion, if the caller and/or the callee register temporarily in other network(s). In this case the hosting MNO can profit from collecting termination rates of the call. On one hand, MNOs can monetize some of their available network resources by attracting more users to use their services, while offering a lower than the usual price in case of low network load. According to the analysis [23], MNOs should increase their focus on new research suggestions, as the proportion of total retail telecoms revenue stemming from their current mobile services is expected to drop over the next five years. On the other hand, MNO subscribers can benefit from lower service charges and/or better Quality-of-Service (QoS) agreements. Despite economical benefits mentioned, an automatic and on-demand MNO selection mechanism can minimize the non-ionizing radiation of the device, especially by each time selecting the MNO with the stronger signal strength, as recently proposed by [28].

Another benefit of the MNO selection flexibility, that an automatic and on-demand MNO selection mechanism supports, can be introduced by regulating authorities. This can enforce the competition in the traditionally considered monopoly of the Mobile Termination Rates (MTRs), as it was introduced in previous work on challenging the monopoly of mobile termination charges with an Auction-based Charging and User-centric System (AbaCUS) [35].

A widely used automatic and on-demand MNO selection mechanism should be supported by many mobile devices that can be equally used in almost every MNO across the world. Thus, those devices should be able to operate in multiple 3rd Generation Partnership Project (3GPP) technologies such as 2/3/4G [24] and should have high market penetration. Smart phones fulfill those criteria; according to [32], since 2011 smart phones are the primary customer's choice. Android is one of the most popular platform in the smart phones market [5]. Thus, the decision to implement and evaluate the automatic and on-demand MNO selection mechanism on the Android platform has been taken. Thus, the research question answered in this work here reads as: Is it feasible and efficient to design an automatic and on-demand MNO selection mechanism, which supports the attempt to overcome the mobile termination rates monopoly obstacle? And the engineering question answered concerns the path in which way to implement effectively and efficiently such a prototype in the Android platform.

978-1-4799-0913-1/14/$31.00 ©2014 IEEE

The remainder of this paper is structured as follows. Related work is discussed in Section II, followed in Section III by the design of the automatic and on-demand MNO selection mechanism. Major results of the evaluation in respect to time consumption, energy efficiency, and AbaCUS end-to-end (e2e) calling process duration and signaling messages data consumption are presented in Section IV. Finally, Section V summarizes the paper, draws conclusions, and presents future work.

II. Related Work

The development and the efficiency in terms of time and energy consumption of an automatic and on-demand MNO selection mechanism for smart phones is important not only from a research point of view [35], [28], but also driven by the recent discussion in the Android developers community [27]. Thus, it is essential to invest key effort toward the development of such a mechanism, as well as for the evaluation of the solution implemented.

A. Auction-based Charging and User-centric System

A monopoly is a corporation that is the only seller of a good or a service, and thus it can define the price. However, monopolies can be divided into two categories, the naturally defined and the market-defined monopolies. The power market in many countries is considered to be a natural monopoly [16] and the main reason is that there is usually only one wire reaching each house. Thus, only the company that owns the delivery network can provide power services. The termination service in mobile communication is also considered to be a monopoly [34]. However, this is a market-defined monopoly, since there is no physical limitation (*e.g.*, wires) for reaching a mobile user. Call termination rates of MNOs are heavily regulated by national telecommunications regulation authorities across the world due to the monopolistic characteristics of the call termination service. However, today the technology allows for different charging mechanisms that could overcome the call termination market monopoly obstacle. The AbaCUS charging mechanism [35] proposes that MNOs participate in an auction, where they bid on termination rates per location and per QoS parameters, such as the network access priority and the sound QoS during a call. AbaCUS require computational effort in the terminal device (smart phone), prior to the establishment of a call, to support the flexible termination rates selection. One of this operations is that the callee should switch from one MNO to another, so the caller will benefit from lower termination rates and/or the better QoS. Thus, an automatic and on-demand MNO selection mechanism is needed to support and handle the MNO switching in the application layer.

B. Electrosmog Reduction through Network Selection

Another novel work to make use of an automatic and on-demand network selection is proposed by [28]. A network switching selection model and its algorithms minimizes the non-ionizing radiation of devices during use. The key goal is to minimize the exposure of the mobile user to electromagnetic radiations, while still providing a certain QoS level. Within a proof of concept [28] validated the model and its algorithms. Due to the fact that the Android Application Programming Interface (API) does not provide for a mechanism to force switching from one MNO to another, the user has to manually select a network. This takes a lot of time, because the provided mechanism by the Android platform searches first for all available networks, which is a time consuming operation. This time overhead makes it impractical to apply a MNO selection algorithm. However, the available MNOs in a country are well known and do not change often. Thus, to avoid the MNO searching delay in this work here, this operation is skipped while a MNO is selected. Available MNOs in a location are stored in a list and when needed the respective MNO is selected from that list. Nevertheless periodic updates of that list, *e.g.*, daily, when an application starts, or when a user moves in a predefined area, are essential to ensure that all currently available MNOs are stored in the list. Thus, the proposed MNO selection mechanism in this paper here can be used for the non-ionizing radiation minimization purpose as well. The evaluation of this mechanism, in terms of energy and time consumption per MNO switching can define a threshold of a maximum number of hops allowed in the non-ionizing radiation minimization MNO selection approach, so that the electrosmog reduction approach remains both realistic and energy efficient.

C. Attention (AT) Commands

The Attention (AT) commands interface has been a standard way to access modems as computer peripherals [37]. Generally an AT command consists of three parts. It starts with `AT` followed by a command and ends with the line termination character [15]. There are three different types of AT commands (Test, Read, and Set).

Test commands test the existence of a command and check its range of parameters. The format of those commands is `ATxxx=?`. To get a list of available MNOs the command `AT+COPS=?` has to be sent to the GSM modem. The reply of the GSM modem returns a list of MNOs with the following information: (a) MNO status (0 unknown, 1 available, 2 current, 3 forbidden), (b) MNO short and long alphanumeric name *e.g.*, `Orange CH` or `ORANGE`, and (c) a five digit number that represents the three digits Mobile Country Code (MCC) followed by the two digits Mobile Network Code (MNC), which is the code for the network provider [29].

The read AT commands, as indicated by the name, read the current value of parameters. Set Commands are used to set new parameter values. The AT command interpreter will return `OK` in the case that the command has been successful, otherwise an error or informative result code will be returned. The MNO set AT command reads as `AT+COPS=1,2,"22801"`. In this command, the first integer defines the mode, with five different values (0 automatic, 1 manual, 2 deregister from network, 3 set only, 4 if manual selection fails, automatic mode is entered). The second integer shows out of three possible values that format the MNO is referenced to (0 long for-

mat alphanumeric, 1 short format alphanumeric, 2 numeric). Thus, if the numeric format has been chosen, the last parameter identifying the MNO is the MCC plus the MNC, *e.g.*, 22801 for Swisscom in Switzerland.

There have been many attempts to send AT commands to Android devices, either as peripheral from a computer or directly from the device itself [17]. But not all issues have been solved yet. Within this work here, an attempt to send AT commands as peripheral from a computer to a Samsung Galaxy S II (SGS2) smart phone was done. Furthermore, an attempt to send AT commands from the device itself took place. The attempt to send AT Commands from a computer was successful. Prerequisites were that the correct GSM modem driver of SGS2 was installed on the computer. Afterwards, the modem could be addressed over the correct device port with a Secure Shell (SSH) client. The outcome of it was a successful MNO selection. However, the approach to send the MNO set AT command from the device itself failed. Until this work concluded, there was no documentation of a successful MNO switching solution via AT commands directly sent from an Android device.

III. MNO Selection Mechanism

Besides the public Android Application Programming Interface (API) that is accessible with the Software Development Kit (SDK), there is also an API, which is located in the package `com.android.internal` [36] that is not accessible via the SDK. While developing Android applications the `android.jar` library is referenced. In this library all classes, enumerations, fields, and methods that are marked with the annotation `@hide`, from the `internal` package are removed. When the application is launched on a device the library `framework.jar`, which is equivalent to the `android.jar`, is loaded. However, the `framework.jar` library provides access with Java reflection [18] to all internal API components from the `internal` package.

A. Accessing the Android Internal API

Accessing the internal Android API requires the `android.jar` library to be replaced by the `framework.jar`. This is not immediately working since the Android Developer Tools (ADT) plug-in for the Integrated Development Environment (IDE) Eclipse [14] forbids the usage of any instrument in the `internal` package by adding an access rule to the java build path. Thus, a developer that need to access anything from the internal API has to do the following steps: 1) obtain the original Android framework, 2) create a custom Android framework, and 3) modify the Eclipse access rule.

B. Obtaining the Original Android Framework

There are two different ways to obtain the original Android framework. One approach is to compile an own framework, due to the fact that Android is an open source mobile Operating System (OS) [13]. However, there exists another path for getting the runtime equivalent and being loaded onto the device at `/system/framework`. Within this work the second approach has been chosen, because it is less time consuming. After the `framework.jar` library is downloaded it has to be extracted by the command `jar xf framework.jar`. If the extracted folder does not contain a file `classes.dex` the file `framework.odex` has to be downloaded from the device as well. This file has to be disassembled with `baksmali.jar` [31] by the following command: `java -jar baksmali framework.odex`. If errors occur with the suggestion to download more odex files, missing files have to be downloaded in the same directory with the `framework.odex`. This will generate the Android platform related classes as smali files [7] in a folder named `out`. This folder has to be assembled with the command `java -jar smali out`. The assembled file is named `out.dex` and is equivalent to the file `classes.dex`, which has to be converted to a jar file using a tool called `dex2jar` [11]. The resulting jar file has to be extracted with the command `jar xf framework.jar`. The extracted folder contains all class files of the `internal` package in the folder corresponding to the package name.

C. Creating a Customized Android Framework

To access the internal API in an IDE, such as Eclipse, a custom framework has to be created, which contains classes and methods of the internal package. To create the custom framework the Android's SDK `android.jar` has to be extracted. This file is located in the Android's SDK installation folder in `SDK/platforms/android-X/android.jar`, where X is the API Level that is targeted at to be customized [2], *e.g.*, level 15 for Android 4.0.4. All files being extracted from the original Android framework have to be copied into the previously extracted folder overwriting already existing files. All files in this folder have to be compressed again into `android.jar` and added to the build path. All methods of the `internal` package are now accessible.

The original Android framework library has to be replaced with the custom platform by replacing the original `android.jar` with the one created. Alternatively, the framework created can be added as new platform. To add a new platform, the entire folder of the original platform has to be copied. The original `android.jar` has to be replaced with the custom one. To distinguish this custom framework from the original one, a custom name and a custom API level has to be provided by adapting the file `build.prop` in the platform folder. The value under the entry `ro.build.version.sdk` has to be replaced by a desired number, which represents the API level. The `ro.build.version.release` value should be expanded with `.extended` to indicate that this is a customized platform.

D. Modifying the Eclipse Access Rule

The last hurdle is to modify the Eclipse access rule that prohibits the use of the internal API. There are different possible ways to achieve this. The first approach is to modify the ADT source code and build it, which has not been investigated within this work. Another way is to modify the ADT's bytecode. Therefore, the content of the

file com.android.ide.eclipse.adt_*.jar, which is located in the folder plugins of the Eclipse installation has to be extracted. The value of * in the file name depends on the ADT version. In the subfolder com/android/ide/eclipse/adt/internal/project of the extracted folder the file AndroidClasspathContainerInitializer.class has to be opened in an editor that supports non-printable characters. The string com/android/internal/ needs to be replaced with another string such as com/android/internax/**. In turn, the folder has to be compressed with the same name as before. It has to be ensured that the internal root folder of the archive is the same as the original one, otherwise Eclipse will not recognize it. Finally, the archived folder has to be renamed to *.jar and the original ADT jar file has to be replaced with the new one. After restarting Eclipse the internal API is accessible. Another approach worked successfully only with ADT version 21 and 22: create a new access rule that allows to use classes out of the package com/android/internal/**. Since the access rule in the subentry android.jar cannot be modified, a new access rule should be created directly below the android platform.

E. Invoking the MNO Selection Mechanism

Although the Android API does not provide any method to change the MNO, the class GSMPhone exists within the Android 4.0.4 source code [6]. This class contains a public method selectNetworkManually. This is part of the internal package and can be used for the purpose of the automatic and on-demand MNO selection mechanism. The class PhoneFactory provides a method to get different kinds of phone objects. To instantiate a GSMPhone object the method getGsmPhone has to be invoked [9]. Afterwards, the method selectNetworkManually can be invoked by the GSMPhone object with the required parameters OperatorInfo and a Message. OperatorInfo contains the information about the MNO to select. This includes the operator information, similar to the AT commands case, as alpha numeric long, alpha numeric short, and numeric. Here a selection could be performed, when a new OperatorInfo object with a correct MNO was created. Other values can be null or empty. Before this mechanism is usable, two further steps have to be performed: (1) run the application with a different shared user ID and (2) run the application under the phone process. To prevent a SecurityException that is thrown, when protected intents [3] are sent by the methods invoked, the application has to run either with the system user ID: android:sharedUserId="android.uid.system" or with the shared user ID android:sharedUserId="android.uid.phone" [9]. This ID has to be set in the AndroidManifest.xml within the manifest-tag.

Additionally, the application has to run in the process android:process="com.android.phone" to ensure that the invocation of getGsmPhone is allowed. This attribute has to be added into the application-tag. Due to the reason that the shared user ID is used by more than one applications, all applications have to be signed with the same certificate [21]. Thus, the application has to be signed with the system signature key. To get such a key is to run a custom Read Only Memory (ROM), which provides these certificates [10], *e.g.*, Cyanogen-Mod. The process of signing an application according to [4] is the following: First, the application has to be exported as an unsigned application package. Second, the platform.x509.pem and platform.pk8 have to be downloaded. Third, these files have to be put into the same folder as the application to be signed.

Before the application is sent to the device, it has to be signed with the tool jarsigner and the command: java -jar signapk.jar platform.x509.pem platform.pk8 Application.apk signedApp.apk. Finally, to sent the application on the device, the partition has to be remounted from a superuser with read-write privileges.

IV. Evaluation of the MNO Selection Mechanism

Energy is a critical resource in mobile communications. Furthermore, long delays are critical for services with a real-time network access, such as phone call establishment, and they may affect the end-user's Quality-of-Experience (QoE)? Thus, having an on-demand and automatic MNO selection mechanism that consumes a lot of energy, or takes a lot of time to switch between MNOs will be practically unusable. Considering that, an evaluation in respect to the energy consumption and the time needed between MNOs switching has been performed.

A realistic evaluation of the MNO selection mechanism has to elaborate multiple successful SIM card registrations between various MNOs in the same location. However, the majority of MNOs accept in their networks only SIM cards issued by them or their roaming partners. Here, a set-up where MNOs accept a SIM card on their network was mandatory for the MNO switching process. Thus, two prepaid SIM cards issued by Mobile Virtual Network Operators (MVNO) have been used. The SIM cards selected have been chosen with the criterion that the registration is possible, while in roaming with every MNO in Switzerland (Orange, Sunrise, and Swisscom).

A. Time Consumption between MNO Switching

There exists a certain SIM card registration time overhead with a SIM card in roaming due to the fact that the local MNO needs time to authorize the foreign SIM card before accepting it within its network. To minimize the authentication overhead, the two SIM cards that have been used were registered to each available MNO prior to the measurements. Thus, a record for each SIM card would be present already during MNO switching measurements, especially in every Visitor Locator Register (VLR) of each MNO.

Thus, the registration time measured had the minimum possible authentication overhead. However, since these SIM cards used during the measurements were prepaid, the available balance authorization overhead could neither be avoided nor estimated. Furthermore, no guarantee

was given that the registration process of the SIM card in roaming to a local MNO was performed with high priority.

The MNO look-up process might take more than 30 s [20]. Thus, the MNO list has been gathered once and their constant availability during the measurements was assumed. The MNO switching average time between the three available MNOs in Switzerland took place for the following two scenarios: (a) when the device was placed in an urban area inside a building and (b) when the device was moving on a train from Zürich to Lucerne. For a comprehensive test the switching took place between all possible MNO pairs. Thus, one test step consisted of 6 switches. This test was repeated 100 times, which led to 600 hops in total. Finally, the time needed for the entire test was measured and the average time needed per case was calculated.

To examine, if the MNO switching time is correlated to the signal strength the cell id and the corresponding signal strength have been measured in the device once, immediately after the MNO hop, and then 4 more times after every 0.5 s. The reason why the signal strength has been captured 5 times is to confirm that its strength was stable. The corresponding cell id has been tracked to make sure that a possible signal strength oscillation is not due to a change to a different cell. According to [12] the relation between the Received Signal Strength Indication (RSSI) values and the signal strength in dBm is outlined in Table 1.

TABLE 1: SIGNAL STRENGTH VALUES

RSSI value	Signal strength [dBm]
0	-113 dBm or less
1	-111 dBm
2-30	-109 dBm to -53 dBm
31	-51 dBm or greater
99	not known or not detectable

In those tests implemented the device always tries to register to an MNO until the attempt is successful. The measurements for case (a) was executed 6 times during a weekday in the following time frames (8:00, 10:30, 12:00, 14:00, 17:00, 2:00 hours). These time frames have been selected so that these measurements undertaken are spread during the day, when the network state (*e.g.*, the network load) changes between rush hours. Thus, the data of the MNO selection consists of total 6 times 100 hops collected in different hours, concluding a total number of 3600 hops. The data of the MNO selection for case (b) consists of a total of 100 hops per MNO pair, reaching a total number of 600 hops, resulting in MNO switching times as shown in Figure 1.

The first MNO, which appears in the caption below the first set of bars, is always the MNO, where the device was registered first, and the second MNO is the one that has been switched to. Each bar corresponds to all switches performed, from the indicated MNO to another. Error bars indicate the standard deviation of all measurements. However, there is a minimum time needed to complete the 6-step SIM network registration process [26]. Thus, the assumption that the minimum MNO switching time cannot be in practise lower than the lowest value measured in this work (4.36 s) has been taken. Left bars present the average switching time between the MNOs at the same location; right bars present the average switching time between the same MNOs while moving. The last set of bars presents the mean MNO switching time for all cases (a) and (b) in summary. The large standard deviation results from large maximum values (cf. Figure 2). Due to the unstable availability of MNOs while moving on a train the maximum MNO switching values appear to be much higher compared to the experiments at the same location is some cases. Furthermore, the MNO selection time shows a quite unstable behavior in some of the cases, which might be related to specific MNO's infrastructure configurations or the current capacity of the connected cell. However, the average MNO switching time is similar in both cases showing that the MNO selection mechanism performs well in every scenario.

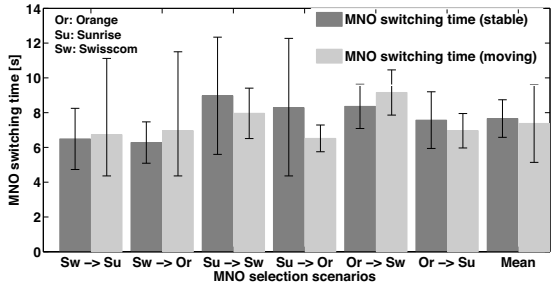

Fig. 1: Switching Time between MNOs

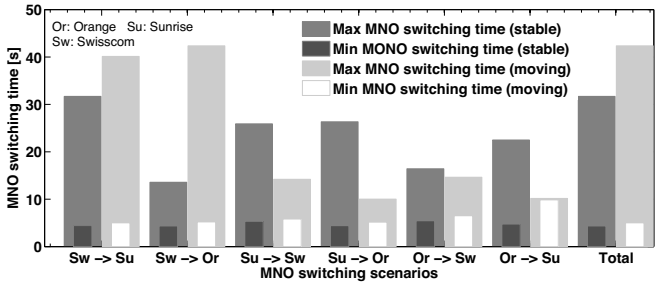

Fig. 2: Min and Max Values for the MNO Switching Time

Figure 3 correlates the MNO switching time with respective RSSI values (12, 8, 5, 3, and 1) of the new MNO. These numbers on each bar indicate the total number of times that the respective signal strength occurred out of the total test hops for scenarios (a) on the left bar and (b) on the right bar. Thus, error-bars represent the standard deviation of these measurements and they are larger in cases, where the respective signal strength has been captured only a few times. It can be seen that the signal strength is not affecting significantly the total switching time from one MNO to another. However, more measurements in a controlled environment, where more values per signal strength are captured, can lead to more accurate conclusions.

Finally, Figure 4 presents the correlation of the MNO switching time for case (a) in those 6 time-slots that the experiment occurred in. It can be seen that the minimum MNO switching time is stable in every time-slot. However, the average and the maximum values are higher in the morning and early in the evening. A possible explanation of this is that the MNO's available capacity in a cell is lower when people are moving in the morning or after lunch to their offices. Furthermore, Figure 5 shows how many times the MNO switching time exceeds 10 s each time-slot. The considerably higher values around 14:00 hours are most likely due to the high network usage at that time.

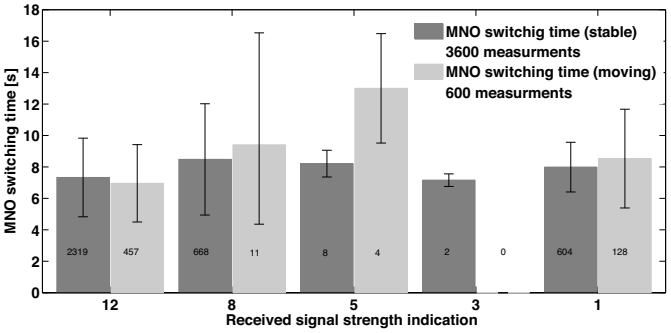

Fig. 3: MNO Switching Time and Signal Strength Correlation

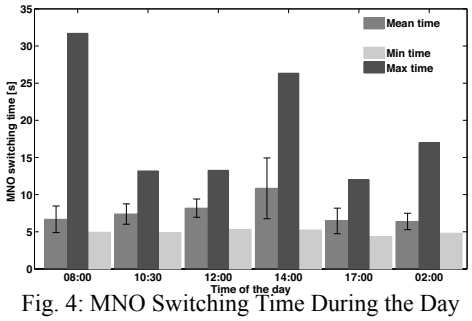

Fig. 4: MNO Switching Time During the Day

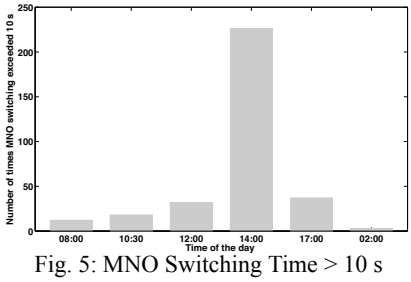

Fig. 5: MNO Switching Time > 10 s

B. MNO Switching Energy Consumption

The power consumption is critical in a mobile system. If a mechanism would absorb a large amount of available energy resources within a few network hops, the MNO switching mechanism would not be usable in practice. Hence, a detailed evaluation of the power consumption has been made. To measure the power consumption, the battery level was determined before the test run and after the test had been performed according [25]. The difference of these levels lead to the final battery consumption in percentage of the battery energy. The assumption is that the battery health is in ideal condition. This assumption is appropriate, because the device of those measurements and its battery was new and experiments were performed in an ideal temperature for the battery [33]. This procedure was applied, since currently no Android application exists, which can measure the real battery capacity, or no application is in place, which measures the consumed energy per application accurately.

$$E = 6,11Wh \cdot 3600s = 21996J \quad (1)$$

$$P_{tot} = \frac{E(J)) \cdot Battery_{used}}{t_{experiment}(s)} = \frac{0,14 \cdot 21996J}{5696s} = 0,5406W \quad (2)$$

The total energy of the battery of the device used is 6.11 Wh according to the manufacturer. Thus, the total energy that a new battery can produce is 21996 J (cf. Equation 1). During the test the display of the device remained turned off, as well as every irrelevant to the experiment process was disabled. In the test case (a), where the location was stable, the measured battery consumption was 14% and the total duration of the measurements was 5696 s. This corresponds to the energy consumption of 3079.44 J. To reach the total power for the MNO switching mechanism the consumed energy has to be divided by the total experiment time. However, this calculation includes also the energy needed for capturing the signal strength and the cell ID five times for each MNO switching measurement. This overhead does not affect significantly the result concerning the MNO switching mechanism, because the energy consumed on this process is small compared to the energy demanded by the MNO switching process. Thus, the results show a total 0.5406 W consumed power for the MNO switching mechanism (cf. Equation 2). The same test has been performed in test case (b), while moving from Zürich to Lucerne by train. The test lasted 7404 s and 22% of the battery was consumed. This corresponds to a total power consumption of 0.6536 W. By comparing these values of both tests, it is evident that in the case the mobile device is not moving the power consumption of the MNO switching mechanism appears to be approximately 17.3% lower than the power consumption, when device is moving, most likely due to the handover energy consumption that can not be isolated. The MNO selection mechanism power value is comparable to the power consumption of the talk mode in 3G networks, which is calculated considering manufacturer's maximum stand-by and talk-time in 2G and 3G networks, as shown within Table 2.

TABLE 2: MOBILE DEVICE CHARACTERISTICS

Consumption	2G max talk time [h]	2G power [W]	3G max talk time [h]	3G power [W]
Voice service	18.33	0.3333	8.67	0.705
Stand-by	710.00	0.0086	610.00	0.010

TABLE 3: ABACUS DATA VOLUME CONSUMPTIONS

Activity	Received [B]	Sent [B]
Register GCM	128	457
Service request	358 (1)	796 (4)
Change MNO	256 (2)	276 (3)
Unregister GCM	128	437

C. AbaCUS E2E Time & Data Consumption

To estimate how long it takes to be established a phone call in AbaCUS, the Au^2 server, which return the MNO that will terminate a call, has been mocked in a local network. A phone call according to the AbaCUS protocol (cf. Figure 6) took place. The total time from the initiation of the call until the callee's phone was ringing was measured. The Au^2 mock server simulated that Swisscom always wins the AbaCUS auction since the AbaCUS distributed auction mechanism described at [35] is not implemented yet. The test call has been done 30 times. Ten times the callee's device had to switch from Orange to Swisscom and ten times it had to switch from Sunrise to Swisscom. Additionally, in ten more cases the time was measured when the callee's MNO did not have to change, because it already was Swisscom. These results are summarized in Figure 7. The different bars indicate the average time consumed for a call termination, where error bars are representing the standard deviation of all measurements. The average time in the case where no MNO change happened is still comparable with the normal dialing case where AbaCUS is not involved. Thus, is shown that for the AbaCUS protocol the average time of a call establishment process mainly depends on the dialing time. The difference of the calling time that a MNO switching is involved, to cases where the MNO has not been switched, corresponds to the average MNO selection time that has been evaluated in Subsection A above. The first MNO in the caption below the bars indicates to which MNO the callee's device was registered before the call. The second MNO is the winning one, which the callee's device had to switch to. The third bar means that the callee's device was already registered to the winning MNO, and the last bar shows the average time when dialing with the traditional dialer without AbaCUS.

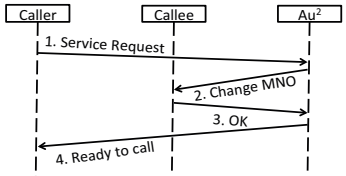

Fig. 6: AbaCUS Protocol (Service Request)

The data consumption of the AbaCUS signaling messages [35], [20] has been measured with an application called Bandwidth Monitor [8]. Table 3 summarizes the data consumption for every AbaCUS signaling message that has been measured. A service request is always sent from a caller to the Au^2 server. The structure of the messages from the caller to the Au^2 server is known, because it is sent as JSON [19]. However, on the other direction the structure is not known because it is handled by Google Cloud Messaging (GCM). The registration process to the GCM server for the push notifications is necessary the first time the application starts as well when the SIM card is changed. To deactivate this push notifications the application has to be unregistered from the GCM server. All numbers before those message names in Figure 6 corresponds to those numbers in brackets in Table 3.

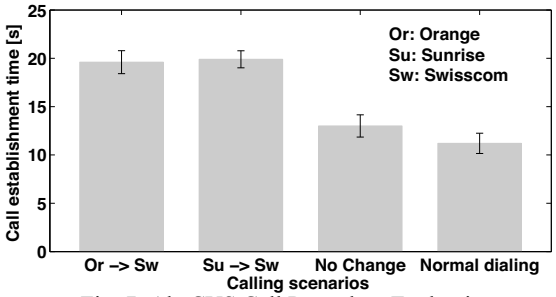

Fig. 7: AbaCUS Call Procedure Evaluation

V. Summary, Conclusions and Future Work

In this work a prototype of an automatic and on-demand MNO selection mechanism for the Android platform had been designed and implemented. The evaluation of the mechanism showed that the time consumption as well as the communication to the server was negligibly low. For both scenarios (the stable mobile user in an urban location or moving in a train) it was shown that the MNO switching time is independent of those MNOs involved and on average it is expected to be below 10 s. Secondly, the power consumption of an MNO switching is in the same dimension of the power needed, when a phone call is performed (cf. Table 4). To demonstrate key requirements, an Android application has been implemented, which makes use of the MNO selection mechanism. To determine the data AbaCUS messages require, an evaluation of the data consumption was performed and results showed that there is only low overhead in the AbaCUS protocol. Thus, an automatic and on-demand MNO selection mechanism is proven to be a realistic and feasible requirement in case that new MNO selection policies need to be applied in the future.

Concluding, the existing knowledge on how to use the internal Android API has been combined to gain access to methods in connection with the GSM modem, which the open Android API does not provide. Thus, an automatic and on-demand MNO selection mechanism has been designed, implemented, and tested successfully. However, this workaround is not the best method, since the internal API is not listed and also may change in the future. Nevertheless, this work showed that such a mechanism is doable and realistic from an energy and time perspective, especially when it is compared to other type of services, such as the traditional phone calls. Thus, the source code of the developed mechanism in this work can be found at [1] and [20]. However, since such a mechanism could be used in the mobile termination rates monopoly break or it could open the window for additional services, such as on-demand QoS-guaranteed services, an automatic and on-demand MNO selection mechanism should be published in the open API by all smart phone vendors.

Finally, in the future all AbaCUS messages will be encrypted and further evaluation concerning the time, the power demand, and the data consumption of the AbaCUS protocol will be done, using additional and different SIM cards (*e.g.*, post-paid ones) and performing measurements in additional settings with other MNOs.

TABLE 4: POWER CONSUMPTION EVALUATION

Process	Power [W]
Talk 3G	0.7050
MNO selection moving	0.6536
MNO selection stable	0.5406
Talk 2G	0.3333

ACKNOWLEDGEMENTS

This work was supported partially by the SmartenIT and the FLAMINGO projects, funded by the EU FP7 Program under Contract No. FP7-2012-ICT-317846 and No. FP7-2012-ICT-318488, respectively.

REFERENCES

[1] AbaCUS Web page, URL: http://www.abacusproject.eu/, Visited in August 2013.
[2] Android API Level, URL: http://developer.android.com/guide/topics/manifest/uses-sdk-element.html#ApiLevels, Visited in August 2013.
[3] Android Intents, URL: http://developer.android.com/reference/android/content/Intent.html, Visited in Aug. 2013.
[4] Android your (my) way, URL: http://omri.org.il/2012/05/25/android-your-my-way/, Visited in July 2013.
[5] Android market share, URL: http://www.ibtimes.com/android-gains-smartphone-os-market-80-share-q2-ios-falls-behind-will-ios-7-revive-apples-os-fortunes, Visited in August 2013.
[6] Android Source Code, URL: http://grepcode.com/snapshot/repository.grepcode.com/java/ext/com.google.android/android/4.0.4_r2.1/, Visited in July 2013.
[7] Backsmali/Smali Manager, URL: http://forum.xda-developers.com/showthread.php?t=2311766, Visited in August 2013.
[8] Bandwidth Monitor, URL: https://play.google.com/store/apps/details?id=org.network&hl=de, Visited in August 2013.
[9] Can a telephony.Phone object be instantiated through the sdk?, URL: http://stackoverflow.com/questions/2143754/can-a-telephony-phone-object-be-instantiated-through-the-sdk?lq=1, Visited in July 2013.
[10] CyanogenMod System Signature, URL: https://github.com/CyanogenMod/android_build/tree/gingerbread/target/product/security, Visited in July 2013.
[11] Dex2jar, URL: http://code.google.com/p/dex2jar/downloads/list, Visited in August 2013.
[12] Digital Cellular Telecommunications System (Phase 2+); Universal Mobile Telecommunications System (UMTS); AT Command Set for User Equipment (UE) (3GPP TS 27.007 version 8.5.0 Release 8), October 2008.
[13] Downloading and Building Android, URL: http://source.android.com/source/building.html, Visited in August 2013.
[14] Eclipse IDE, URL: http://www.eclipse.org/, Visited in August 2013.
[15] 3rd Generation Partnership Project (3GPP), Technical Specification Group Terminals, AT command set for User Equipment (UE) (Release 6), June 2003.
[16] J. D. Gwartney, R. L. Stroup, R. S. Sobel and D. A. Macpherson, "Microeconomics: Private and Public Choice, 14th Edition", Chapter 11 "Characteristics of a Monopoly", Cengage Learning, January 2012.
[17] How to talk to the Modem with AT commands, URL: http://forum.xda-developers.com/showthread.php?t=1471241, Visited in July 2013.
[18] Java Reflection, URL: http://docs.oracle.com/javase/tutorial/reflect/, Visited in August 2013.
[19] JSON, URL: http://www.json.org, Visited on August 2013.
[20] S. Liniger, "Implementation of an automatic, on-demand Mobile Network Operator (MNO) selection mechanism on Android devices", Bachelor thesis, Communication Systems Group (CSG), Computer Science Department (IFI), University of Zurich (UZH), Zürich, Switzerland, August 2013.
[21] Manifest, URL: http://developer.android.com/guide/topics/manifest/manifest-element.html, Visited in July 2013.
[22] MNO directory, URL: http://www.mnodirectory.com/europe-subs.html, Visited in August 2013.
[23] MNOs must reform tariffs, URL: http://www.eurocomms.com/industry-news/49-online-press/9218-mobile-operators-must-reform-tariffs-to-make-data-switch, Visited in August 2013.
[24] Mobile communication technologies per country, URL: http://www.gsmarena.com/network-bands.php3, Visited in August 2013.
[25] Monitoring the Battery Level and Charging State, URL: http://developer.android.com/training/monitoring-device-state/battery-monitoring.html, Visited in August 2013.
[26] K. Pahlavan, P. Krishnamurthy, "Principles of wireless networks", Chapter 7, paragraph 7.3, Prentice Hall PTR, 2002.
[27] Programatically Connecting to Another Network Operators, URL: http://stackoverflow.com/questions/2373727/programatically-connecting-to-another-network-opertors, Visited in August 2013.
[28] J. M. Seigneur, X. Titi, T. Maliki, "Towards Mobile/Wearable Device Electrosmog Reduction through Careful Network Selection", 1st Augmented Human International Conference (AH'10), Megève, France, April 2010.
[29] J. Schiller: Mobile Communications; 2nd Edition, Pearson Education, Harlow, U.K., 2003.
[30] J. Schiller: Solution Manual for Mobile Communications; Second Edition, Freie Universität Berlin, Berlin, Germany.
[31] Smali/Baksmali, URL: https://code.google.com/p/smali/, Visited in August 2013.
[32] Smartphones penetration, URL: http://www.nielsen.com/us/en/newswire/2010/smartphones-to-overtake-feature-phones-in-u-s-by-2011.html, Visited in August 2013.
[33] K. Takeno, M. Ichimura, K. Takano, J. Yamaki, "Influence of cycle capacity deterioration and storage capacity deterioration on Li-ion batteries used in mobile phones", Journal of Power Sources, Vol. 142, No. 1–2, March 2005, pp 298-305.
[34] Telecompetition, "Chapter 5. Call termination", URL: http://www.kilpailuvirasto.fi/tiedostot/telecompetition.pdf, Visited in August 2013.
[35] C. Tsiaras, B. Stiller, "Challenging the Monopoly of Mobile Termination Charges with an Auction-based Charging and User-centric System (AbaCUS)", Networked Systems (NetSys 2013), Stuttgart, Germany, March 2013, pp 110-117.
[36] Using Internal (com.android.internal) and Hidden (@hide) APIs [Part 1, Introduction], URL: https://devmaze.wordpress.com/2011/01/18/using-com-android-internal-part-1-introduction/, Visited in July 2013.
[37] T. Vaattovaara, "Analysis of Modem Integration in Open Source Smartphone Platforms", Master Thesis, Oulu University of Applied Sciences, Finland, March 2011.