Acta Universitatis Sapientiae
Electrical and Mechanical Engineering, 2 (2010) 114-122

Localization of the Mobile Calls Based on SS7 Information and Using Web Mapping Service

Virgil CAZACU [1], Laura COBÂRZAN [2], Dan ROBU [3], Florin SANDU [4]

[1] BitDefender, Bucharest, Romania, e-mail: virgil.cazacu@gmail.com
[2] Softvision, Cluj-Napoca, Romania, e-mail: laura.cobarzan@gmail.com
[3] Siemens Program and System Engineering, Brasov, Romania, e-mail: dan.robu@siemens.com
[4] Faculty of Electrical Engineering & Computer Science, "Transilvania" University, Brasov, Romania, e-mail: sandu@unitbv.ro

Manuscript received October 01, 2010; revised October 20, 2010.

Abstract: Localization of the calls is a topic that has been coming up even from the early time of the telephony. Calls made from mobile phones were even more interested to be localized due to their mobility. This paper presents a localization solution that uses information from the mobile network, being a technical solution that ensures the acquisition of the localization information of the calls from the terminals in the mobile network and which is delivering this data to a localization server. The localization solution that is presented has three major features: receiving calls' information from mobile networks and obtaining the localization information from the Signaling System #7(SS7); data processing from signaling frame and IP-transmitting of this information to a localization server; visualization of the call location on the map. Due to client-server architecture, users of the system can access calls locations using digital maps.

Keywords: Localization, mobile networks, service, client-server, integration.

1. Introduction

Localization of the calls is useful not only from the legal point of view but also in case of emergencies as is for example the usage of the short number 112 or 911. In this case, the localization of the person who is in possible danger is vital.

Using SS7 localization approach has drawbacks which are treated in the presented solution: each mobile phone service provider supplies the localization information within the Initial Address Message (IAM) field of the ISDN User Part (ISUP) protocol from SS7 frame in its' own specific format [1]. Thus, the solution offers the possibility to configure the necessary parameters, depending on the place of deployment.

From the design point of view, the solution ensures the service of acquirement of the localization information for the terminals in the mobile networks and it is delivering this information to a localization server for calls that are selected to be localized. The call processing, localization information extraction and delivering these on the interface to the localization server is performed in almost-real time, delays appearing if the load of the system is high. This solution is adaptable with minimal costs for future changes of the architecture. These changes might include resizing the necessary input/output traffic and the modification of the field from the SS7 signaling frame in which the localization information is transmitted by using parameterized components.

SS7 localization data is sent using “Cell ID” type from ISUP protocol, in the IAM message, “Location number” field and/or “Called number address” field [1].

The next table presents an example of localization data format that is specific for each mobile service provider.

Table 1: Localization data format.

Network code (e.g.72,74 or 4072, 4074)	Services bit (reserved)	Location area code	Cell ID
2-4 digits	1 digit	5 digits	5 digits

Based on these frames, each mobile operator maintains a database with geographic information that can offer information about the caller position based on the positioning string. The database structure is different, according to the telephony provider and contains the equivalent geographical coordinates for the above data from SS7 ISUP frame as it is shown as an example in *Table 2*.

Table 2: Geographical coordinates database structure.

Cell code	District	City	Street	Lat (Grade, Minutes, Seconds)	Long (Grade, Minutes, Seconds)	Azimuth	BSC	LAC
Specific Code	String	String	String	Int (6 digits)	Int (6 digits)	Int (3 digits)	Specific Code	Int (4 digits)

The location of these databases is defined by each mobile operator, which also manages and maintains it. These databases should be interfaced with a solution like the one presented in this paper.

The general solution of the architecture is presented in *Fig.1*.

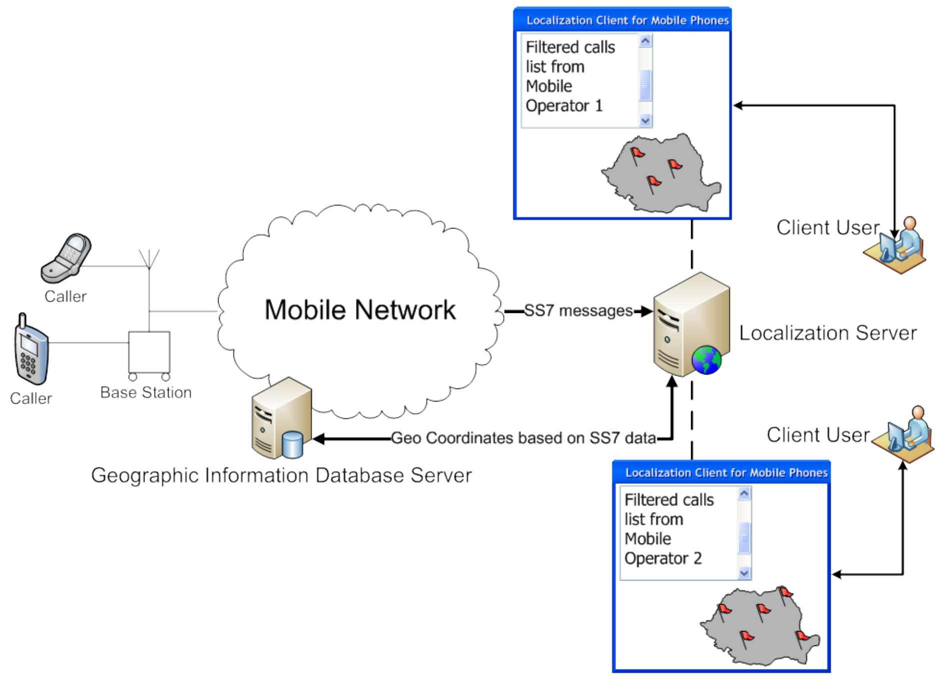

Localization Client for Mobile Phones
Filtered calls list from Mobile Operator 1
Client User
Caller
Caller
Base Station
Mobile Network
SS7 messages
Localization Server
Geo Coordinates based on SS7 data
Client User
Geographic Information Database Server
Localization Client for Mobile Phones
Filtered calls list from Mobile Operator 2

Figure 1: Overview of the solution architecture.

The solution, as the picture above shows, is divided into four modules and it is targeted to be used in operational centers that can coordinate emergency activities:

- Extracting Localization String modules from the Localization Server presumes the definition of rules for parsing SS7 ISUP information. The main service of the server is to define interaction protocols as well as Geo Information Database Server interaction.
- Geographic Information Database Server is maintaining the geo-coordination of the radio cells.
- Client side module is handling communication between Graphical User Interface (GUI) module and Localization Server.
- GUI module is handling specific interface functions and the digital maps using web mapping service available on the market at the solution's implementation time.

2. Description of the main components

This section presents the technical implementation of the solution modules using as examples the case when mobile users are using the Emergency Service 112 and are customers of one of the Romanian mobile operators.

A. Localization Server

In this module, there are two major functionalities: SS7 frame parsing and communication protocols between the client and the Geo Database Server.

The parsing module consists of two parts, one dealing with the SS7 Integrated Services Digital Network User Part (ISUP) communication, while the other being responsible for the protocol message parsing. It is out of the paper's scope to detail the SS7 communication between our solution and the mobile network. The technical approach taken is to use the JAIN ISUP API that gives the possibility to exchange ISUP messages in the form of Java Event Objects [2], [3].

One rule for parsing SS7 information is the fact that independently of the mobile operator, SS7 frames are in standard format and the relevant parameters for our solution can be found under the Initial Address heading. The relevant parameters are presented in the *Table 3*.

Table 3: IAM parameters used for localization.

Parameter Name	**Explanation**
Calling party number	Nature of address: either National or International. Calling address signal: the telephone number of the caller party (with a 2 digit prefix for international calls).
Called party number	Called address signals. For Emergency cases, the called telephone number is 112.
Location number	The localization string, that contains all the localization information for the given provider.
Cell ID	The mobile operator internal ID for the radio cell where the call is made from.

The phone numbers are received without the prefix digits, so in this implementation the "Calling party number" parameter is taken into account. For national calls a 0 digit and for international calls two 0 digits are inserted at the beginning of the caller number. Also, to determine from which mobile operator the call is performed and knowing that every telephone number begins for example with 07XY, where depending on the XY digits, the solution can extract the provider of the call based on a table correspondence and on interrogating the portability server, if available.

In the implementation of the module, the extracted information is stored in an object called SS7Object with fields like String callingNumber, String calledNumber, String Nature, String LocalizationString, Date dateCreated, String Provider. SS7Objects are sent to the Localization Server module for further processing.

The communication protocol with the client uses sockets and when new localization objects are received from the SS7 parsing module, the server will send the object to the client in order to use it on the GUI. After sending the localization object, the server is waiting for an answer from the client. If the client does not confirm the reception of the object in the previously defined time frame, the localization server will resend this information. The Localization objects that are not confirmed are maintained in a waiting list. When the localization server receives a message from GUI/Google Maps, it will delete the corresponding object from the "waiting list", meaning that it will not wait for the confirmation for that object.

The communication protocol between Localization Server and the Geo Information Database Server is done by calling the getCoordinatesBy LocalizationString (localization_string) method which takes a string parameter, representing the localization string and as returned value, an object which contains the coordinates of the area from which the call was made. The coordinates will be the latitude and the longitude, each one containing 3 fields: degrees, minutes and seconds. The calling of the class is done using Remote Method Invocation (RMI).

B. Geographical Information Database Server

As it was mentioned earlier in the paper, this server has to be located at each mobile operator since it contains internal information about the place where radio cells are deployed from geographical point of view. For completeness of the solution description, the RMI Database Server will be presented, that has several classes in order to extract the coordinates from the local database.

The Coordinate class is common with the RMI client, the *CoordinateInterface* class which contains the remote method and the *CoordinateImplementation* class, which implements the remote method as shown below:

public Coordinate getCoordinatesByLocalizationString(String localization_string)

The Provider class has a static method String getProvider(String localization_str), which returns the provider, based on the localization string and the ExtractCoordinate class has a static method that returns the coordinates, based on the provider and the localization string, as it can be seen next capture.

public static Coordinate getCoordinatesByProvider(String provider, String localization_str)

The ExtractFromDatabase class contains one static method for each provider, to extract the coordinates from the local database, based on the localization string, as shown below:

public static Coordinate ExtractVodafone(String localization_string)

A database structure example for this server is presented in *Fig. 2*.

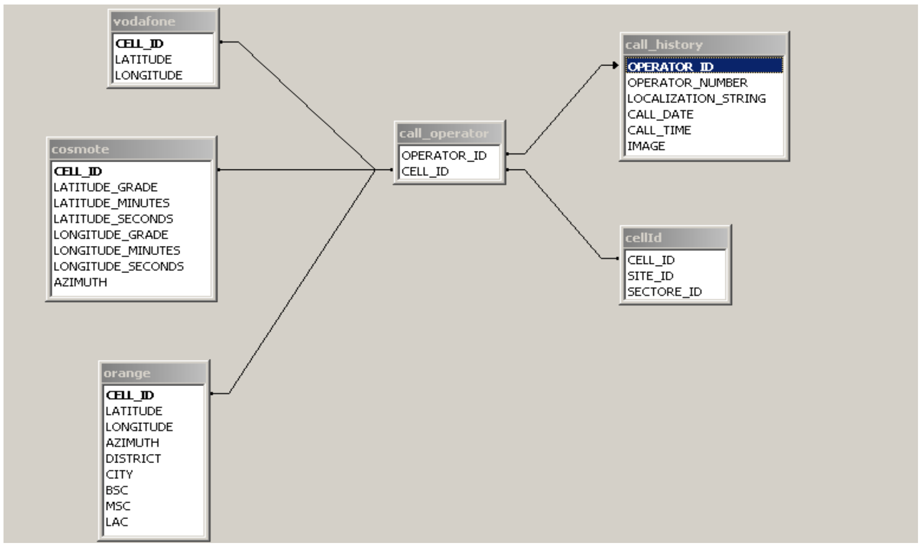


Figure 2: Structure example of geo coordinates database.

C. Client side including GUI

As the communication between the client and the Localization Server is detailed in section B, this part is focused on the usage of web mapping service and user interface.

The graphical user interface presented in this paper has a proof of concept oriented design. The GUI of the solution is composed of two frames: one frame with 4 tabs: View Calls, View Archive, Options and Help. The other frame is displaying the digital map with all its options.

Only one tab is presented in this paper, the View Calls tab which contains the recent calls information in a list. The call information contains the exact time of the call, the caller number, the provider and the location of the call as it is received from the Location Server. Each call has its own checkbox, which will specify if the call was processed or not. When a call is selected in the list, the application is marking automatically the location of the call in the map frame. More calls can be selected simultaneously, so the map can be marked in more locations. Calls that are checked (processed) are deleted from the list and the marks from the map disappear.

In order to integrate a digital map into the solution, Google Maps API was chosen due to several considerations [5].

Google Static Maps API embeds a Google Maps image without requiring JavaScript but the problem is that it returns the map as an image (GIF, PNG or JPEG) in response to a HTTP request via a URL. This way, the benefits of the zoom and navigation facilities disappear.

JXMapViewer embeds mapping abilities into Java application, but at the solution's development time it was not possible to use it with Google Maps or Yahoo since there were legal restrictions.

One other strong reason why the Google Maps API was chosen for integrating the web mapping service was the possibility to control the zoom and navigation features from the application's back-end.

Since Google Maps API uses JavaScript, the *JWebBrowser* class from the chrriis.dj.nativeswing.components package has been used in the development of the Java client application; this offers the possibility to have a native web browser component in the application [4]. Because the client application has to be operating system independent, the web browser component was developed to use the Mozilla engine.

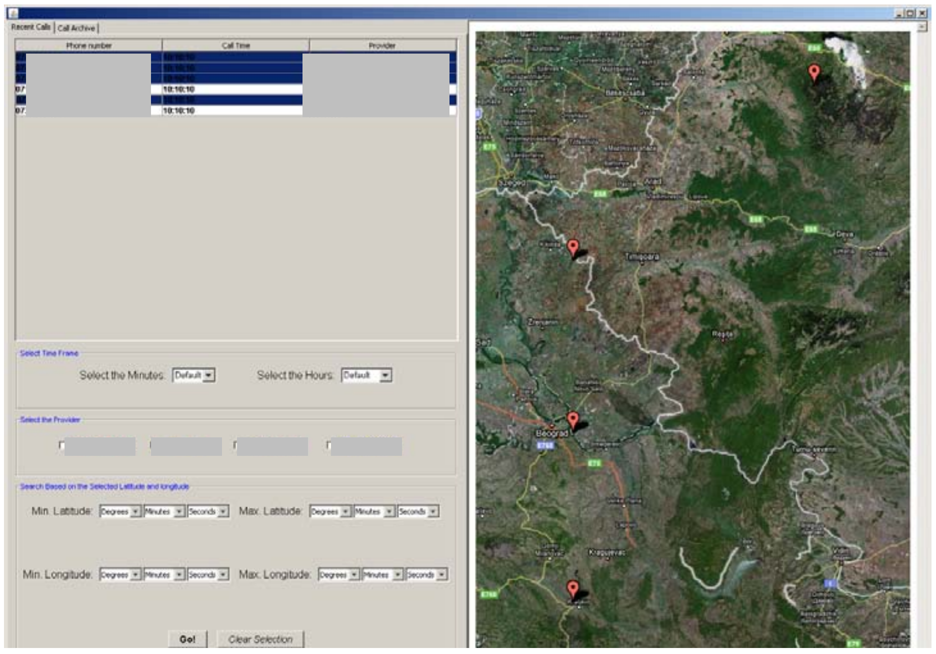
```
NSOption opt = new NSOption(JWebBrowser.useXULRunnerRuntime());
web_browser = new JWebBrowser(opt);
JWebBrowser.useXULRunnerRuntime();
```

The digital map from Google is loaded using the following code line [5].

```
web_browser.navigate(gmapfilelocation.getAbsolutePath());
```

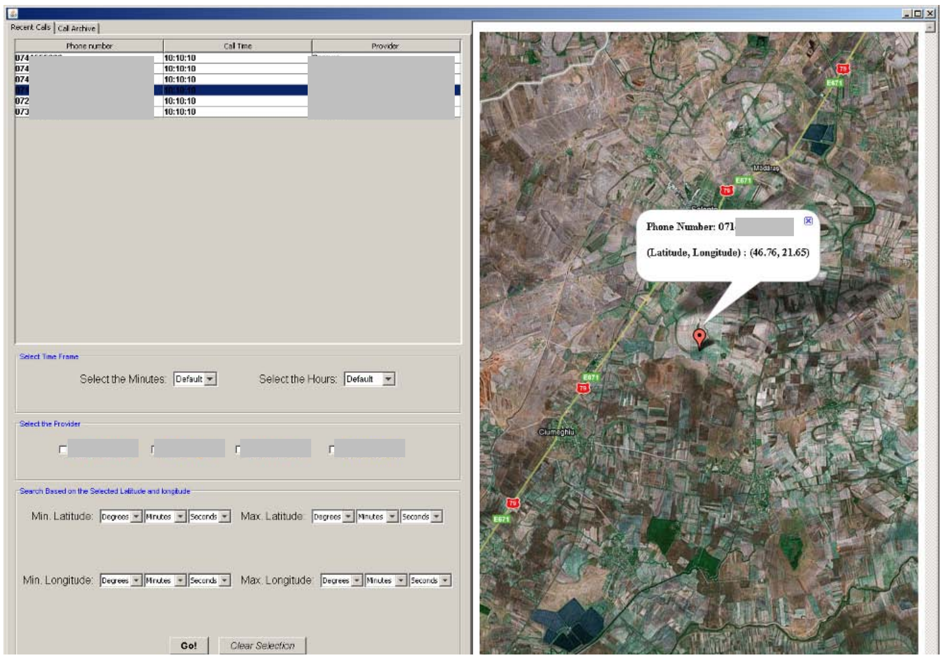
The parameter *gmapfilelocation* points to the file containing the script which loads the map.

In *Fig. 3*, the client GUI is shown together with the calls markers, each marker descriptor containing a string defining the location to place the marker and the visual attributes to use when displaying the mark.



Phone number	Call Time	Provider
[redacted]	10:10:10	[redacted]
[redacted]	10:10:10	[redacted]
07[redacted]	10:10:10	[redacted]
[redacted]	10:10:10	[redacted]
07[redacted]	10:10:10	[redacted]

Figure 3: Calls markers on the map.



Phone number	Call Time	Provider
074[redacted]	10:10:10	[redacted]
074[redacted]	10:10:10	[redacted]
074[redacted]	10:10:10	[redacted]
[redacted]	10:10:10	[redacted]
072[redacted]	10:10:10	[redacted]
073[redacted]	10:10:10	[redacted]

Phone Number: 071[redacted]

(Latitude, Longitude) : (46.76, 21.65)

Figure 4: Zoom on caller location.

Figure 4 shows a case when one call is selected from the list and automatically the application zooms in on the location where the caller is positioned. If another call is selected, so that two calls are on the map, the application automatically zooms out exactly as it is necessary to display both callers on the map.

3. Conclusion

The solution of call localization presented in this paper is still under development since topics like high degree of availability or the ability to work in load-balanced and failed-over conditions between locations are not implemented. The application's architecture has been implemented by the authors of the paper and solution allows further improvements, in order to enable features like accepted input traffic of a high number of simultaneous voice calls to be implemented as easy as possible.

But the goal of the research, at least in this phase, was achieved, since the usage of a web mapping service for calls location has been demonstrated by the solution presented in this paper.

Acknowledgements

The authors wish to thank their colleagues who contributed with their effort to achieve the results presented in this paper, especially the colleagues located at the Cluj-Napoca Siemens PSE site.

References

[1] Dryburgh, L., Hewett, J., "Signaling System No. 7 (SS7/C7): protocol, architecture, and services", Cisco Press, 2005.

[2] Jepsen, T. C., Anjum, F., "Java in telecommunications: solutions for next generation networks", John Wiley & Sons, 2001.

[3] *** http://jcp.org/en/jsr/summary?id=ISUP.

[4] *** http://djproject.sourceforge.net/ns/documentation/javadoc/index.html.

[5] *** http://code.google.com/apis/maps/documentation/reference.html.