We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800
Open access books available

122,000
International authors and editors

135M
Downloads

Our authors are among the

154
Countries delivered to

TOP 1%
most cited scientists

12.2%
Contributors from top 500 universities

CLARIVATE ANALYTICS BOOK CITATION INDEX INDEXED

WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?
Contact book.department@intechopen.com

Numbers displayed above are based on latest data collected.
For more information visit www.intechopen.com

Chapter 2

Prepaid Voice Services Based on OpenBTS Platform

Ladislav Behan, Lukas Orcik, Filip Rezac,
Ivan Baronak and Jerry Chun Wei Lin

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.71099

Abstract

This article describes the design and implementation of prepaid voice services based on OpenBTS platform. By using various programming languages and open-source software tools, we can integrate prepaid voice services with this system, so its functionality is resembled as much as possible the operation of traditional GSM network provider. This article also provides description of how customers will approach their billing services, how they will access their accounts and pay their invoices.

Keywords: GSM, OpenBTS, Asterisk, smqueue, Subscriber Registry, SIP, IAX, USRP, C++, PHP, trunk, VoIP

1. Introduction

Recently, GSM system is increasingly attracting the attention of the open-source community. Software implementation of the traditional GSM network would allow it to operate at much lower costs and would provide easier control over the entire system. The GSM network architecture is a remarkable piece of technology that many technicians developed for a very long time. It is very robust and also scalable but on the other side, quite inflexible and too expensive. Many open-source projects were built in order to address this. For example, Osmocom, OpenLTE, and YateBTS and Open Base Transceiver Station (OpenBTS) [1, 2]. Each project was built with different targets and architectures, tackling weak spots of traditional network in its own way.

Linux application OpenBTS (Open Base Transceiver Station) is one of the projects that allows user to create GSM network based on a software implementation. Due to economic software design and proper selection of powerful amplifiers, it allows user to install and operate a low-capacity GSM network at one-tenth of the cost of current technologies only. This project implements the GSM air interface (Um) that allows cellular handsets to be used as SIP endpoints. It provides Um interface by Universal Software Radio Peripheral (USRP) [3]. There are

still many areas on the Earth that do not have mobile network coverage or even telephone lines at home. But, they do have an Internet connection via satellite or long-haul WiFi. OpenBTS is able to convert and distribute this Internet connection as a mobile network across a large geographic region. This solution is suitable for building GSM in extreme conditions and the achieved PESQ quality [4] of such solution is presented in several papers [5, 6]. Any mobile phone connected to this network can transmit basic data, use voice services or send text messages. The combination of OpenBTS and USRP changes the way we should think about mobile networks. This technology allows user to build a complex radio network purely based on open-source software tools. OpenBTS is written in a C++ programming language and this application implements the whole GSM stack. OpenBTS is just a software. You can make it do whatever you would like, so OpenBTS network's capabilities can be enhanced with a non-difficult software update. The mobile network is finally open for innovation and anyone is able to build experimental cellular network now [7, 8].

There is no need for any configuration changes on the mobile stations, because the radio interface of OpenBTS network is equivalent to mentioned GSM network. The core of an OpenBTS network is composed of open protocols and IP is used as its transport protocol. Many software projects already exist and implement these open protocols, but there were some new components developed in order to provide a functionality to link the GSM and IP technology. Additionally, there are various applications providing efficient tools for the investigation of the GSM security issues [9–11].

The rest of this paper is organized as follows: In Section 2, the OpenBTS project is described in detail. Section 3 shows the design and implementation of accounting services integrated to our custom cellular network. Method of charging for voice is presented in Section 4. In Section 5, the application of control of exceeding the limit of prepaid services is presented. Section 6 describes how text messages are monitored in our network. Section 7 concludes the paper.

2. OpenBTS and its architecture

OpenBTS implements a complete GSM stack for voice, SMS and allows calls between registered mobile stations within created network and also between different network providers. These networks can be used to support true fixed-mobile convergence, bring coverage to remote areas or experiment and innovate within the cellular network itself. Because OpenBTS converts both cellular signaling and media directly to SIP [12] and RTP [13], the integration environment is quite familiar [14].

As mentioned earlier, OpenBTS itself is written in the programming language C++ and uses Asterisk PBX to place calls. Asterisk also allows this system to connect to various private or public IP networks. Mobile stations connected to OpenBTS network can reach each other even if the system is not connected to the Internet, but reaching someone outside the network requires an Internet connection. This new "hybrid" architecture is illustrated in **Figure 1** [15].

The crucial element of the whole OpenBTS architecture is a product of the Ettus Research, USRP. This is a relatively inexpensive hardware that is easily adaptable to GSM transmitter.

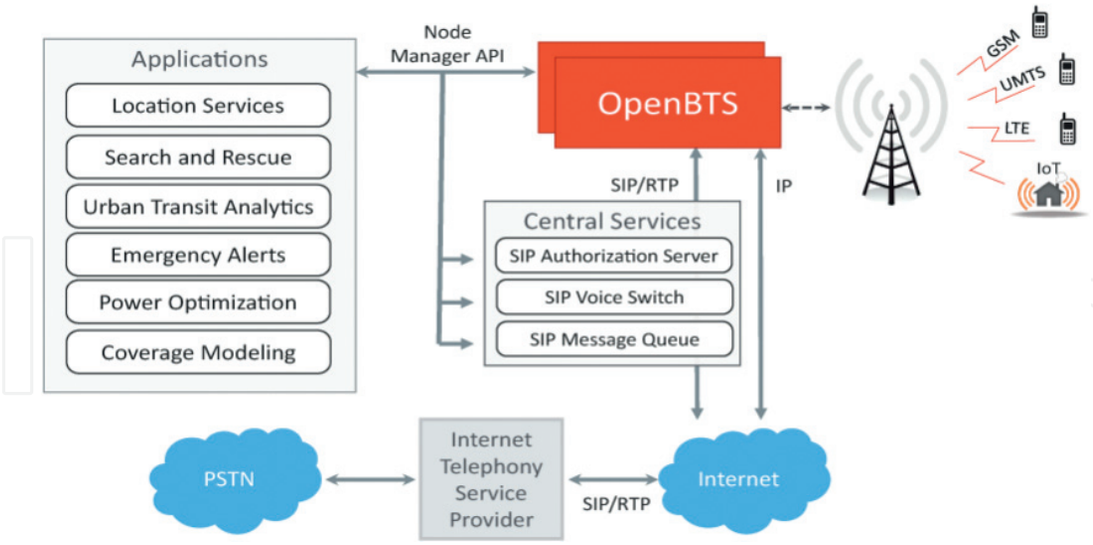

Figure 1. OpenBTS architecture [15].

USRP system consists of a motherboard which can be easily extended by the additional cards to provide and transmit signals with different frequency bands. It also consists of an USB interface, through which it communicates with the computer and a programmable FPGA. Motherboard contains A/D and D/A converters, processor interface and the controller power system to generate and synchronize clocks and FPGA. USRP needs UHD for a proper functionality. UHD is an open-source driver that is compatible with all operating systems and it is possible to use with LabView, GNU Radio or OpenBTS (**Figure 2**).

Sipauthserve is an application that implements Subscriber Registry, the database of subscriber information that takes place at both the Asterisk SIP registry and the GSM Home Location Register (HLR) that can be found in a conventional GSM network. The delivery of each text message in OpenBTS network depends on a store-and-forward facility. This facility is provided

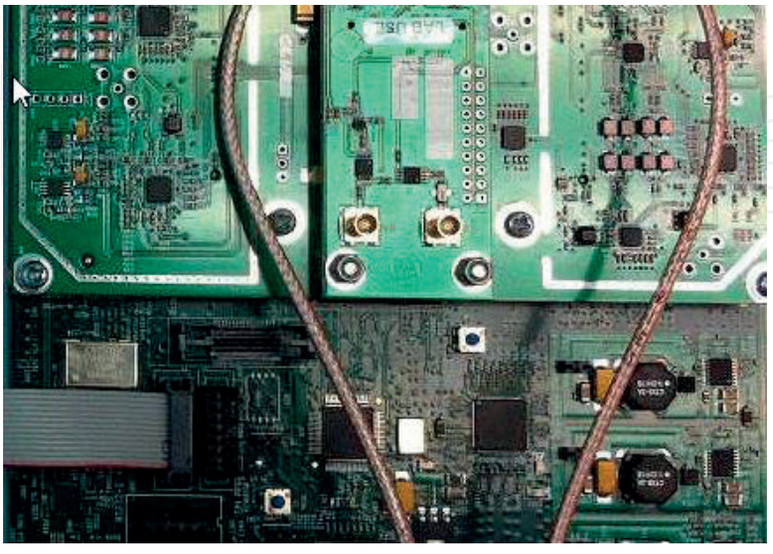

Figure 2. OpenBTS USRP N210 from Ettus Research [9].

by smqueue. The core of smqueue is a queue of messages that are waiting for delivery. They wait in this queue, potentially during multiple delivery attempts, until they are successfully delivered or until messages are determined to be undeliverable. The function of smqueue server is similar to email server [1].

3. Design and implementation of accounting services

Figure 3 shows connection scheme of OpenBTS network. Asterisk, OpenBTS, smqueue and sipauthserve are running on PC with the assigned IP address (158.196.229.242). A network switch connects PC, USRP N210 and IP phone together. In this topology, Asterisk is connected to two other Asterisk servers via IAX and SIP trunk. There are softphones connected to both of these servers.

Now we can connect any mobile station (MS) to our created network. If the connection was successfully established, the IMSI number of the phone SIM card is stored in the TMSITable database. We created a C++ program that continuously checks this database for a new IMSI records. If the new IMSI appears, then this program will check whether the record exists in sip_buddies table. If sip_buddies table does not contain this IMSI number, then program will generate a phone number, inserts a record to this sqlite3 database and modifies the configuration files of Asterisk.

As you can see in **Figure 4**, if MS successfully connects to OpenBTS network, it will obtain a welcome message with information about its generated phone number and IMSI number. With these numbers user is able to login onto his account at the service provider's website where he should activate voice services. Every generated phone number or IMSI number is unique so there is no possibility to duplicate a data. After user's successful logging, the welcome page should appear with a main menu where user can choose from following items: Services, Statistics, Summary and Contact. After opening the Services tab, users are offered to be able to activate various services. They are able to choose either one of the credit services or activate one of the fixed payment tariffs (**Figure 5**).

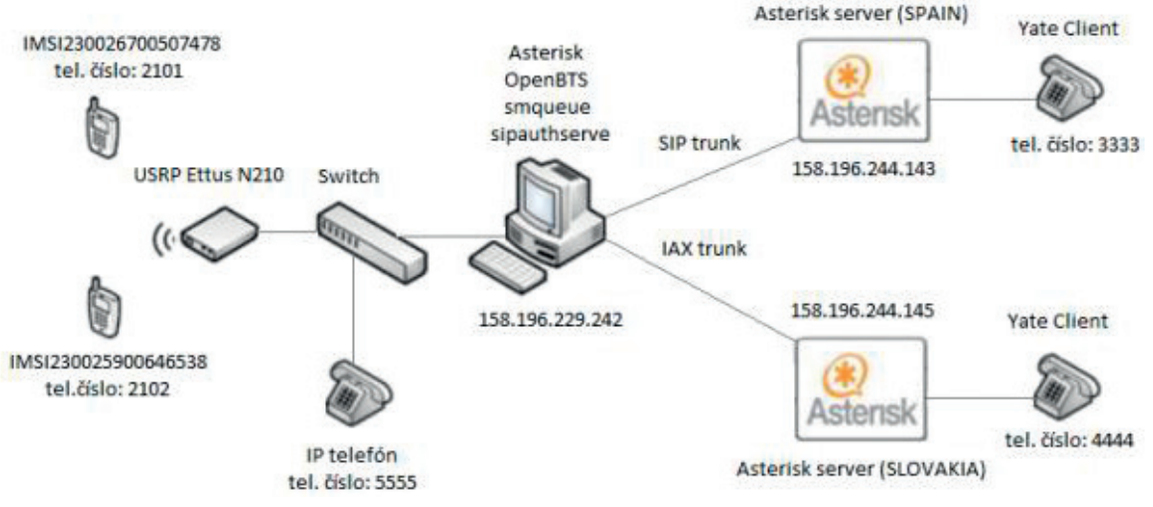

Figure 3. Connection scheme of OpenBTS network.

```
INFO:    Searching for new IMSI...
SUCCESS: New IMSI found =>IMSI230024701231666 !!!
INFO:    Opening database...OK
INFO:    IMSI not used, creating new record.
INFO:    Inserting to db: id(2), number(2101), imsi(IMSI230024701231666)
SUCCESS: Inserting to dialdata_table...OK
SUCCESS: Inserting to SIP_BUDDIES...OK
SUCCESS: /etc/asterisk/sip.conf updated.
SUCCESS: /etc/asterisk/extensions.conf updated.
INFO:    Closing DB...OK
INFO:    Welcome message was successfully sent !!

INFO:    Searching for new IMSI...
```

Figure 4. Automatic IMSI registration in OpenBTS network.

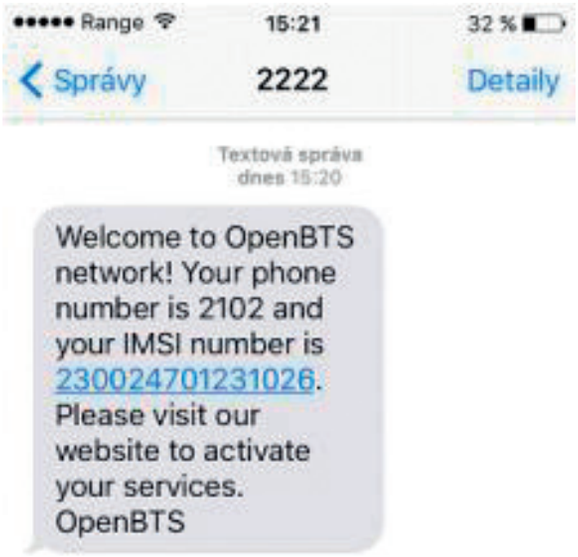

Figure 5. OpenBTS welcome message.

Each button containing a prepaid voice service is connected to the PayPal Internet-based payment system. After choosing preferred service, user will be redirected to PayPal website, where he will be asked to pay the amount for the specific tariff. In the Summary tab, see **Figure 6**, there is possibility to review all calls that have been made retrospectively. This table shows to which extensions user was calling and to which network area this number belongs. Then he can view the total duration of the call with exact start and end time of the call. Last item is showing the amount charged for this call. User is also able to download this entire summary report as a PDF file. Files are created by the FPDF tool that is a PHP class which allows to generate PDF files.

The Statistics tab consists of individual statistics. For example, as we can see in **Figure 7**, the user is able to view the number of calls he made in each month. The number of text messages he sent is shown in **Figure 8** and the total amount of money paid for voice services is shown in **Figure 9**. These graphs are generated by object-oriented PHP library JPGraph that is processing user's data retrieved from a SQL database.

OpenBTS telefónica

SUMMARY OF CALLS

Id	User	Country	Called	Start	End	Duration (sec)	Price(CZK)
1	2101	CzechRep	2102	16:51:14	16:52:22	8	4
2	2101	CzechRep	2102	16:53:28	16:54:20	52	26
3	2101	SPAIN	3333	17:00:18	17:00:56	38	38
4	2101	SPAIN	3333	17:02:16	17:02:48	32	16
5	2101	SPAIN	3333	17:05:01	17:06:23	82	41
6	2101	IPPHONE	5555	14:25:31	14:26:17	46	35
7	2101	IPPHONE	5555	18:41:12	18:43:11	59	44
8	2101	SLOVAKIA	4444	20:21:33	20:22:49	76	152

Figure 6. Summary of calls table.

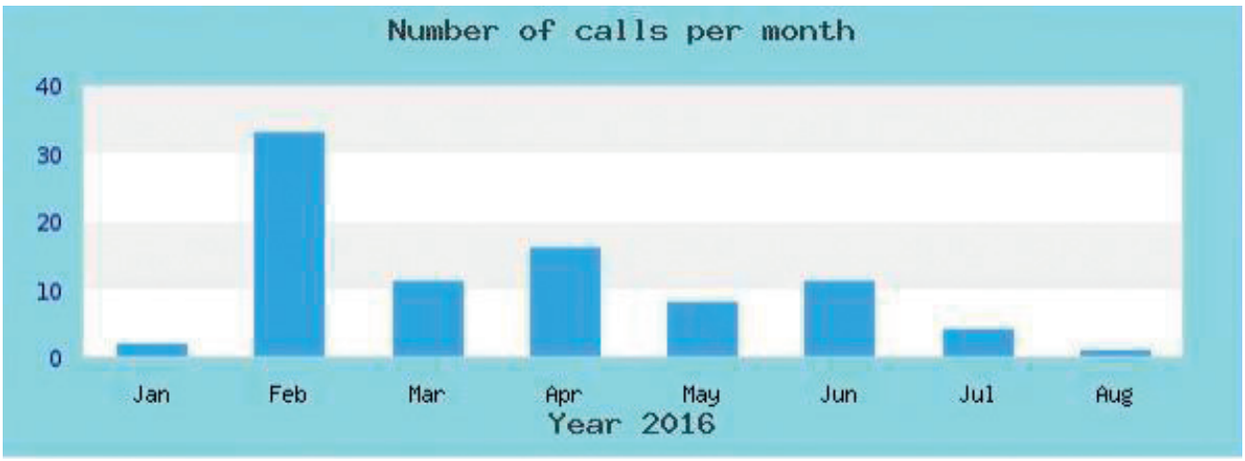

Figure 7. Number of calls made per month.

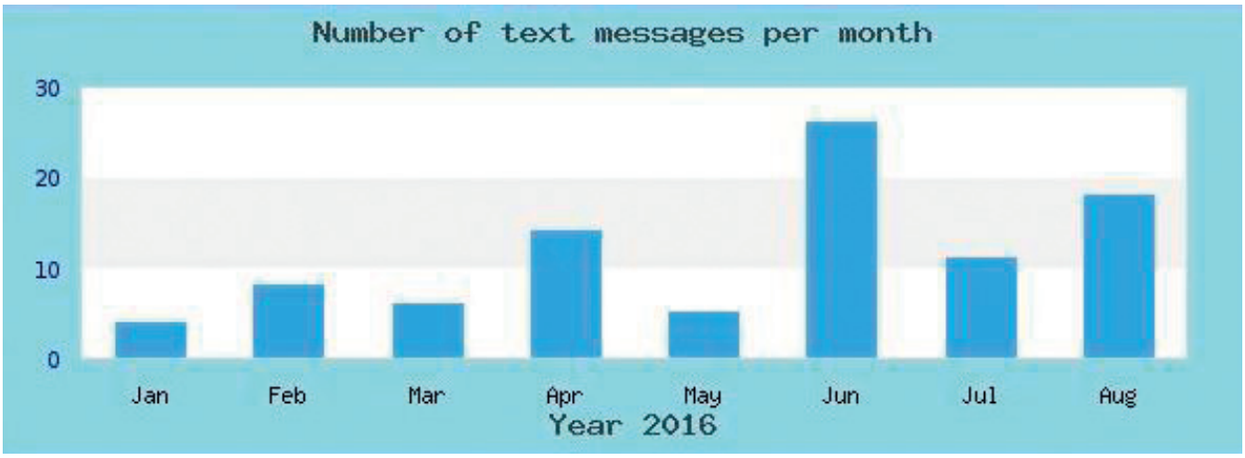

Figure 8. Number of text messages sent per month.

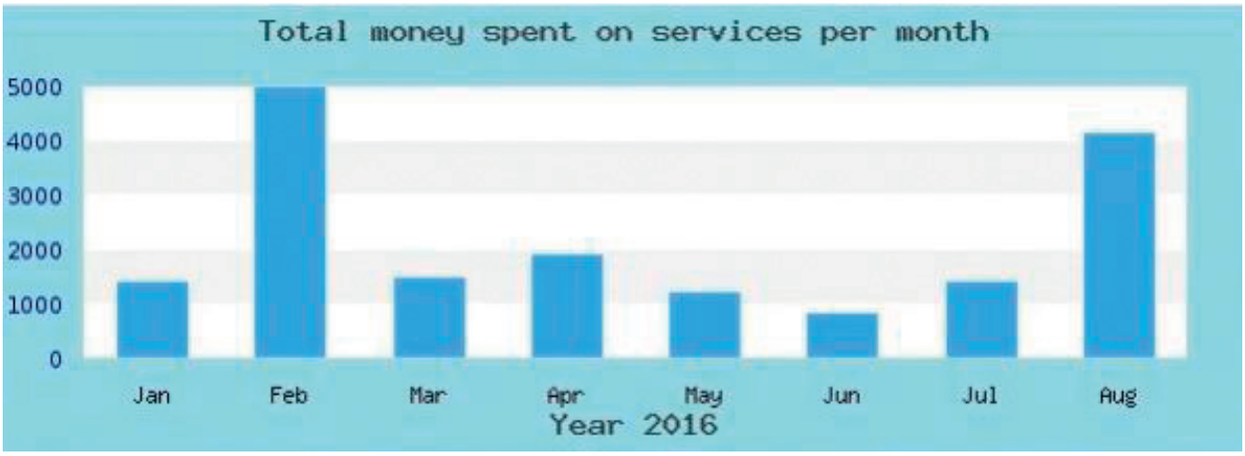

Figure 9. Total money spent on service per month.

4. Method of charging for voice services in the OpenBTS network

Phone calls in OpenBTS network are routed via the Asterisk PBX that stores detailed records related to these calls in the Master.csv file. Asterisk stores these calling data records (CDR) to this csv file after every call is processed. CDR records consist of very important data for our billing system implementation. For example, the phone number of the subscriber originating the call, the phone number receiving the call, the call duration or the starting time of the call (date and time).

We implemented a C++ application that parses these CDR records from Master.csv file and stores them to various databases used in our billing system. This application is continuously checking Master.csv file for a new record. If a new CDR record has been added, application

```
SUCCESS: SQLITE DB was successfully OPENED!
SUCCESS: Username found in DB!
INFO   : The call lasted : 68 seconds
INFO   : User -> 2101 has service -> cash
INFO   : Updating credit status...
INFO   : Call was made in Czech Republic, prize -> 34 CZK
INFO   : Updating users table...
SUCCESS: SQLITE DB was successfully UPDATED!
SUCCESS: MYSQL DB was successfully OPENED!
SUCCESS: MYSQL DB was successfully UPDATED!
SUCCESS: MYSQL DB was successfully CLOSED!
SUCCESS: MYSQL DB was successfully OPENED!
SUCCESS: MYSQL DB call data successfully INSERTED!
SUCCESS: MYSQL DB number of calls was UPDATED!
SUCCESS: SQLITE DB was successfully CLOSED!
```

Figure 10. Program flow of cdr_manager.Cpp.

will find out what is the user's activated service. If he has activated one of the tariff services, then the total call time will be subtracted from the rest of his free prepaid minutes. If he has activated the credit service then the resulting sum of the call will be calculated in accordance of where the call was routed (IP phone, SIP/IAX trunk, MS). **Figure 10** shows the flow of cdr_manager application when the new CDR record has been found.

Figure 11 shows that application also checks if the user's credit balance is not too low. If it finds out the remaining credit is under 20 CZK then it will send a notification message to him.

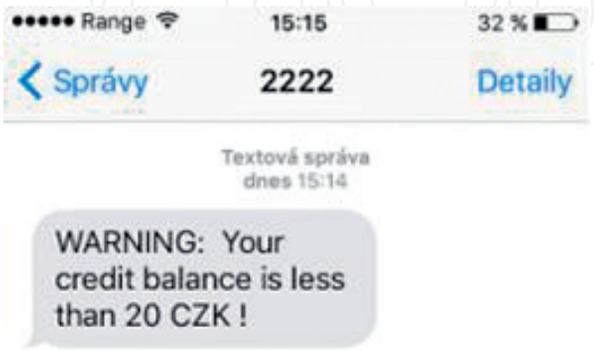

Figure 11. Low credit balance message.

5. Control of exceeding the limit of prepaid services

Another essential part of our billing system is to check if the caller did not exceed his limit of prepaid services during the active phone call. We implemented an application call_manager.cpp that constantly monitors occupancy of Asterisk PBX channels. **Figure 12** shows the flow of this application.

As we have mentioned, the whole process starts with constantly enquiring Asterisk about the state of the call channels. If Asterisk confirms that there has been an active channel found, the call_manager checks if a caller extension has its prepaid service active. If not, the

```
SUCCESS: Database was successfully opened!
SUCCESS: Active channel found!!
INFO   : Active channel is used by -> 2101 <=> 2102
INFO   : 2101 has service -> cash
INFO   : User 2101 is allowed to use this channel for 60 secs
INFO   : Allowed time was exceeded -> terminating active channel!!!
INFO   : Text message was sent to user!
INFO   : Updating credit status for user: 2101
SUCCESS: Updating database..OK!
INFO   : User -> 2101 don't have enough money to make a call!!!
```

Figure 12. Call_manager.cpp flow.

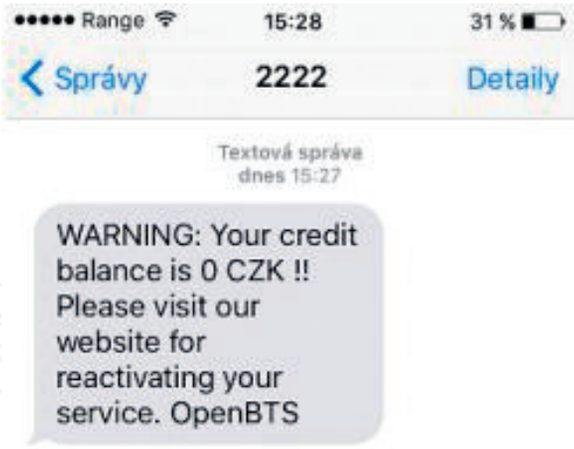

Figure 13. Zero credit text message.

application will terminate this channel immediately and the following text message is send to MS (**Figure 13**).

If the user's account is non-zero then the application will determine whether the call will be routed through SIP/IAX trunk, IP Phone or call will be made on the OpenBTS network. Based on that data our application will determine the time for how long is the caller able to use the voice channel.

In cases when the time limit has exceeded, the voice channel will be terminated immediately, the program opens a database containing information about the caller's prepaid services and set them to zero. At that point, the user will be informed about the zero balance of his account via SMS. He is also invited to visit the service provider's website to reactivate his service, otherwise he will not be able to make calls or send any text messages.

6. Text message billing

Another important part of our billing system is a way to monitor sent text messages. In this case, we implemented application that is parsing messages stored in OpenBTS.log file. This program regularly monitors mentioned file and select only messages with specific content like the following example:

Mar 2 15:03:22 studentPC smqueue: NOTICE 4842:4848 2016-03-02 T15:03:22.1 smqueue.cpp:2455: main_loop: Got SMS rqst qtag '658,054– OBTSbwgdfkphscqborwe' from IMSI230024701231026 $.

The other application sms_manager.cpp then parses this message and picks information about IMSI number. According to this IMSI number, program checks the user's service and its status. If he has activated one of the tariff services then the total free text messages is subtracted by 1, in case of a credit service there is a fee charged for this text message. The program then updates the database that indicates the number of messages sent per month.

```
INFO   : User with imsi number IMSI230024701231026 sent sms!
SUCCESS: SQLITE DB was successfully OPENED!
INFO   : His phone number is -> 2101
INFO   : Updating SQlite table...
SUCCESS: SQLITE DB was successfully UPDATED!
INFO   : Updating MySQL table...
SUCCESS: MYSQL DB number of text messages was successfully UPDATED!
```

Figure 14. Sms_manager.cpp flow.

As mentioned earlier, these statistics are then graphically available on the service provider website (**Figure 14**).

7. Conclusion

In this article, we demonstrated the possibility of creating a billing system working correctly in conjunction with the OpenBTS platform and providing an important part of the mobile operator system based on OpenBTS. First of all, we automate the registration of IMSI numbers, which means that after the phone is registered to the network, the phone number is automatically assigned to it. This author's contribution in community of OpenBTS project. Using various software tools and programming languages, we have implemented our own billing system that we integrated to OpenBTS system and it led to low-cost mobile operator creation. Last that is missing to completeness is data service integration that can be provided by integrating OsmoSGSN a OpenGGSN nodes to our system. OpenBTS system proves that nearly anyone can run a custom GSM network with parts from a home-supply or auto-supply store. The technology required to build this kind of network is no longer far too expensive, nor has a fistful of big companies locked them down. Mobile phone users within this network are able to make calls to each other and if the network is connected to the Internet, they can make calls to people around the world.

Contribution of this paper lies in an experimental development of the solution for mobile operators which is based on the OpenBTS platform with widely available USRP HW and SW tools implemented within this experimental research. The solution can be operated in areas, where mobile operators do not cover, and no mobile infrastructure exists, especially on islands.

Acknowledgements

The research presented in the paper was supported by the SGS grant no. SP2017/174, VSB–Technical University of Ostrava, Czech Republic.

Author details

Ladislav Behan[1]*, Lukas Orcik[1], Filip Rezac[2], Ivan Baronak[2] and Jerry Chun Wei Lin[3]

*Address all correspondence to: ladislav.behan@vsb.cz

1 Faculty of Electrical Engineering and Computer Science, VSB—Technical University of Ostrava, Ostrava, Czech Republic

2 Faculty of Electrical Engineering and Information Technology, Slovak University of Technology, Bratislava, Slovak Republic

3 School of Computer Science and Technology, Harbin Institute of Technology Shenzhen Graduate School, Shenzhen, Guangdong, China

References

[1] Burgess DA, Samra H. The Open BTS Project–An Opensource GSM Base Station. September 2008. [Online]. Available at: http://www.ahzf.de/itstuff/papers/OpenBTSProject.pdf

[2] Mikulec M, Voznak M, Fajkus Z, et al. Building GSM network in extreme conditions. Proceedings of SPIE-The International Society for Optical Engineering. 2015;**9478**: art. no. 94780K

[3] Kemetmuller C, Seeger M, Baier H, Busch C. Manipulating mobile devices with a private GSM base station—A case study. In Proceedings of the 8th International Network Conference (INC 2010), 2010

[4] Rozhon J, Voznak M. Development of a speech quality monitoring tool based on ITU-T P.862. 2011 34th International Conference on Telecommunications and Signal Processing TSP 2011—Proceedings. art. 6043771. 2011. pp. 62–66

[5] Partila P, Kohut M, Voznak, et al. A methodology for measuring voice quality using PESQ and interactive voice response in the GSM channel designed by openBTS. Advances in Electrical and Electronic Engineering. 2013;**11**(5):380-386

[6] Fajkus M, Mikulec M, Voznak M, Tomis M, Fazio P. Speech quality measurement of GSM infrastructure built on USRP N210 and openBTS project, Advances in electrical and. Electronic Engineering. 2014;**12**(4):341-346

[7] Sankhe K, Pradhan C, Kumar S, Murthy GR. Cost effective restoration of wireless connectivity in disaster hit areas using OpenBTS. 11th IEEE India Conference: Emerging Trends and Innovation in Technology, INDICON 2014; 2015. art. no. 7030511

[8] Yuova Kumar S, Saitwal MS, Khan MZA, Desai UB. Cognitive GSM OpenBTS. Proceedings—11th IEEE International Conference on Mobile Ad Hoc and Sensor Systems. MASS; 2014. art. 7035736 2015. p. 529–530

[9] Song Y, Zhou K, Chen X. Fake BTS attacks of GSM system on software radio platform. Journal of Networks. 2012;**7**(2):275-281

[10] Voznak M, Prokes M, Sevcik et al. Vulnerabilities in GSM technology and feasibility of selected attacks. Proceedings of SPIE-The International Society for Optical Engineering, 9456. 2015. art. no. 94560T

[11] Cattaneo G, De Maio G. Ferraro Pertillo U, Security issues and attacks on the GSM standard: A review, Journal of universal computer. Science. 2013;**19**(16):2437-2452

[12] Voznak M, Rozhon J. Approach to stress tests in SIP environment based on marginal analysis. Telecommunication Systems. 2013;**52**(3):1583-1593

[13] Burget R, Komosny D, Ganeshan K. Topology aware feedback transmission for real-time control protocol. Journal of Network and Computer Applications. 2012;**35**(2):723-730

[14] Iedema M. Getting Started with OpenBTS. 2015. Available at: http://openbts.org/site/wp-content/uploads/ebook/Getting_Started_with_OpenBTS_Range_Networks.pdf

[15] OpenBTS. [Online]. Available at: http://openbts.org/about/