The University of Mississippi Undergraduate Research Journal

Volume 2 Article 12

4-1-2017

Open Source Mobile Network Using OpenBTS and USRP 9

Erin Bratu

Bryan Harper

Open Source Mobile Network Using OpenBTS and USRP 9

Erratum
2017-04-01

Open Source Mobile Network Using OpenBTS and USRP

Erin Bratu and Bryan Harper **Advised by Mr. Raviteja Chinnambeti and Dr. Lei Cao**
Department of Electrical Engineering, University of Mississippi, University, MS 38677

***Abstract*— This paper describes the implementation of a GSM (Global System for Mobile) network using a USRP (Universal Software Radio Peripheral) module integrated with OpenBTS (Open Base Transceiver Station). The implementation can perform the same basic functions as a commercial GSM network, including voice, SMS (Short Message Service), MMS (Multimedia Messaging Service), and GPRS (General packet Radio Service). The advantage of this implementation is that a self-contained cellular network can be created with a single computer. This simple network can be extended with multiple nodes to ideally use for situations where mobile communications infrastructure is absent or compromised (e.g., in disaster struck areas).**

I. INTRODUCTION

The creation of affordable, transportable, easily installable, and reliable networks has been a prime motivation in the field of communication. These network features can be realized by integration of hardware and software as a single functioning network unit. Such a network is called a Software Defined Network (SDN). The advantage of using an SDN over traditional networking methods is that much of the configuration can be adjusted as needed, with few to no changes to the existing hardware infrastructure required.

This paper describes the use of USRP as a hardware unit and OpenBTS as a software unit to create a functioning, portable GSM network. This network can be used for Search and Rescue (SAR) missions in disaster struck areas, as a local commercial or non-commercial network in remote areas for communication, as a local network between employees in an industrial zone e.t.c..

All of the above situations call for a dynamic, low-cost mobile network that can be configured as needed. A demonstration of this type of system may be seen in Fig. 1. This particular system uses a computer running Linux Ubuntu 14.04, a USRP B210, two GSM-band antennae, and Android-based mobile phones, as well as the Asterisk and OpenBTS softwares to create the network. The implementation of GSM network using USRP can be found online but never given in detailed description, this paper gives full insight on implementation including the debugging process if any error persists during installation process, as well as presents some field test results.

The rest of the paper is organized as follows. Section II explains the system components and requirements. section III describes the implementation procedure of GSM network. Section IV shows the performance of the system in a practical scenario, and discusses possible ways for network extension. Finally, section V concludes the paper.

Fig. 1. Hardware Setup.

II. SYSTEM AND COMPONENTS

Open Base Station Transceiver

OpenBTS is an open source software, written in C++, that can be used to implement a self-contained cellular network. Peripherals—such as USRP devices—are used as base stations. GSM (Global System for Mobile Communications) devices can be used as mobile stations, as OpenBTS uses the GSM protocol stack [2]. However, core network functions, such as resource management, connection establishment, switching, etc., are performed internally on the computer(s) running the software [3]. The services which can be provided include voice, SMS, MMS, and data.

Asterisk

For long-distance calling functionality we use the software Asterisk. Asterisk is an open source software implementation of a PBX (Private Branch Exchange). In addition to communication within the PBX, Asterisk is capable of connecting with other networks such as the PSTN (Public Switched Telephone network) or VoIP (Voice over Internet Protocol) services [9]. OpenBTS delivers calls via SIP (session initiation protocol) via Asterisk. Without Asterisk or a VoIP soft switch, OpenBTS would require calls to be forwarded through an operator's mobile switching center.

Universal Software Radio Peripheral

USRP is an inexpensive hardware platform for software defined radio. Generally, a USRP device is connected to a host computer by using high-speed links. Host-based software is used to control the USRP in order to transmit

and receive data. All USRP products utilize free, open-source USRP hardware driver (UHD) software package. USRP devices are commonly used with the GNU Radio software suite to create complex software-defined radio systems.

The USRP hardware driver (UHD) is provided by Ettus Research (National instruments subsidiary) for use with the USRP product family. It is supported for use with Linux, MacOS, Windows systems and several frameworks including GNU Radio, LabVIEW, MATLAB and simulink. The UHD functionality can also be directly accessed using UHD API which provides native support for C++. Any other languages that can import C++ functions can also use UHD. For example, this can be used by Python through Simplified Wrapper and Interface Generator (SWIG).

USRP B200/B210 are used in our implementation and their specifications include:

- USB 3.0 interface
- Xilinx Spartan 6 XC6SLX75 FPGA
- A Cypress EZ-USB FX3 High-speed USB 3.0 controller
- Analog Devices AD9361 RFIC
- Coverage from 70 MHz - 6 GHz RF
- Flexible rate 12 bit ADC/DAC
- 1 TX, 1 RX, Half or Full Duplex (B200) / 2 TX, 2 RX, Half or Full Duplex (B210)
- Fully coherent 2×2MIMO capability (B210)
- Up to 56 MHz of real-time bandwidth 1×1
- Up to 32 MHz of real-time bandwidth 2×2 (B210)

III. IMPLEMENTATION

TABLE I
REQUISITE EQUIPMENT FOR OPENBTS IMPLEMENTATION WITH USRP.

Item	Quantitiy
USRP (B210 / B200)	1
Mobile Phone	2
SIM Card	2
Antenna ($900 - 1800$ MHz)	2
Computer (Ubuntu 14.04)	1

Step 1: Install Operating System

This implementation requires that one's operating system can be run using the "Upstart" initialization daemon, as OpenBTS currently does not have support for other systems, such as "systemd" which is a system and session manager for linux. Upstart is available in all Linux Ubuntu distributions beginning with Ubuntu 6.10; however, beginning with Ubuntu 15.04, the default is "systemd", so for this project the Ubuntu 14.04 distribution is used.

One can find instructions on hard drive partitions and dual booting with Windows and Ubuntu here: http://www.tecmint.com/ubuntu-14-04-installation-guide/. Alternatively, if enough RAM is available, one can use VirtualBox: https://www.virtualbox.org/.

Step 2: Install GNU Radio and UHD

Open the terminal window in ubuntu and install GNU Radio along with UHD by running the following script:

```
wget http://www.sbrac.org/files/build-gnuradio &&
    chmod a+x build-gnuradio && ./build-gnuradio
```

For any additional instruction or detail refer to [5].

Step 3: Install Development Packages

It is necessary to install the following prerequisite development packages on your Linux machine:

- autoconf
- libtool
- libosip2-dev
- libortp-dev
- libusb-1.0-0-dev
- g++
- sqlite3
- libsqlite3-dev
- erlang
- libreadline6-dev
- libncurses5-dev
- libuhd-dev
- libuhd003
- -host
- libboost-dev
- bind9
- ntp

Each package can be installed by opening a terminal window and entering the following command:

```
apt-get install [package]
```

Multiple packages can be installed with a single command by listing additional arguments:

```
apt-get install [first] [second] ...
```

Step 4: Update and Install Git

Git is a free and open source distributed version control system for small to very large projects, Run the following scripts on terminal window to install Git which is used by OpenBTS:

```
apt-get install software-properties-common
apt-get install python-software-properties
add-apt-repository ppa:git-core/ppa
apt-get update
apt-get install git
```

Step 5: Install SSH

Register your own SSH keys on Github, use Github registered account for this before entering the following command on terminal window.

```
apt-get install ssh
ssh-keygen -t rsa -C ''name@email.com''
ssh-agent -s
ssh-add    /.ssh/id rsa
xclip -sel clip    /.ssh/id rsa.pub
```

After copying the SSH key paste it into the Github account under the SSH key field and enter the following command in the terminal to connect ssh from Github.

```
ssh -T git@github.com
```

Step 6: Install OpenBTS

Create a directory for installation of OpenBTS:

```
cd dev
```

Download OpenBTS:

```
git clone https://github.com/RangeNetworks/dev.git
./clone.sh
```

Select the version or branch:

```
./switchto.sh master
```

or

```
./switchto.sh 5.0 installation of liba53
cd /src/dev/liba53
```

Install `liba53`:

```
cd /src/dev/liba53
make install
ldconfig
cd ../
```

Install `libcoredumper`

```
cd libcoredumper
./build.sh
dpkg -i *.deb
cd ..
git submodule update init recursive
./NodeManager/install libzmq.sh
```

OpenBTS installation and debugging:

```
cd dev
./build.sh B210
```

The above commands should have installed OpenBTS and built the necessary files but if any error persists one can use the following commands:

```
./switchto.sh 5.0
cd openbts
git checkout master
git pull
git submodule init
git submodule update
cd ..
install libsodium 13 dependency file if prompted
./NodeManager/install libzmq.sh
./build.sh B210
```

The above commands should have resolved most of the installation issues.

Step 7: Install `deb` Packages

A Debian Package (.deb) is an archive file that consists of executable files, libraries and documentation associated with particular suite of a program or group of programs. To install .deb files go to BUILDS folder installed in dev and look for the folder named in the format year-month-date hours-minutes-seconds (ex: 2016-03-1410-47-42) and run the following command:

```
dpkg -i *.deb
```

For further help please refer to [6] and [8]. After the deb installation follow the instructions given in [7] to install Asterisk and FreePBX.After that reboot the computer and test the services using the following commands:

```
start sipauthserve
start smqueue
start openbts
start asterisk
```

The above commands will start the services or displays message "start: Job is already running: openbts" on successful installation and similarly use the following commands to stop the services for further configuration of Openbts and Asterisk.

```
stop sipauthserve
stop smqueue
stop asterisk
```

Step 8: Configuration

Configuring the USRP and OpenBTS:

After the successful installation of software components required to setup the mobile network, the next important task to be performed is to configure the software to initiate the connectivity between GSM mobiles with active SIM cards and USRP.

Note: Connect the USRP along with the two antennas before proceeding further.

The easiest way to manipulate the configuration keys is via the OpenBTS command line interface (CLI) with the following shell command:

```
/OpenBTS/OpenBTSCLI
```

Now configure the GSM band and Absolute Radio Frequency Channel Number (ARFCN) by the following command and customize them if necessary.

```
OpenBTS > config GSM.Radio
GSM.Radio.ARFCNs 1 [default]
GSM.Radio.Band 900 [default]
GSM.Radio.C0 51 [default]
GSM.Radio.MaxExpectedDelaySpread 4 [default]
GSM.Radio.PowerManager.MaxAttenDB 10 [default]
GSM.Radio.PowerManager.MinAttenDB 0 [default]
GSM.Radio.RSSITarget -50 [default]
GSM.Radio.SNRTarget 10 [default]

OpenBTS > config
GSM.Radio.Band 850
GSM.Radio.Band changed from 900 to 850
WARNING: GSM.Radio.C0 (51) falls outside the valid
    range of ARFCNs
128-251 for GSM.Radio.Band (850)
GSM.Radio.Band is static; change takes effect on
    restart
```

If ARFCN falls outside the range please use the following command to get it inside the range.

```
OpenBTS > config
GSM.Radio.C0 166
GSM.Radio.C0 changed from 51 to 166
GSM.Radio.C0 is static; change takes effect on
    restart
```

Configuring the Mobile Phones:

Once the configuration is done proceed with the following instructions to detect the USRP OpenBTS network (often seen as Test PLMN 1-1 or 001005).

1) Launch the Settings application from the Android menu system.
2) Select More.
3) Select Mobile networks.
4) Select Network operators. This may or may not start a search. If it does not, select Search networks.
5) Once the search has finished, a list of available carrier networks is presented.

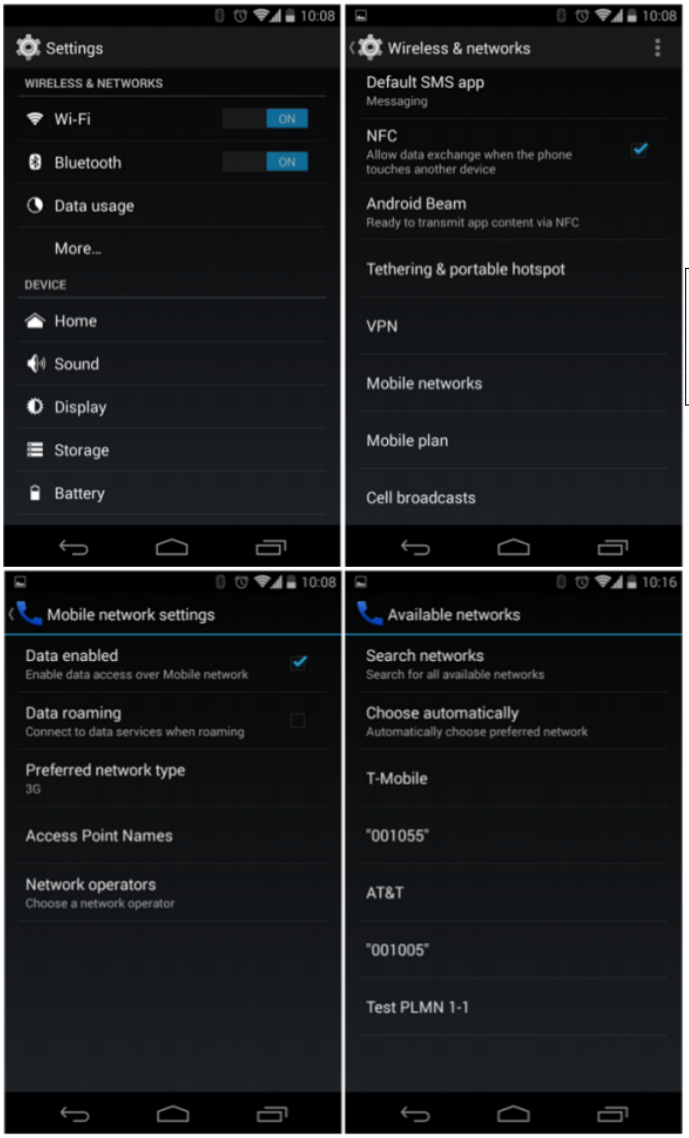

Fig. 2. Manual carrier selection of network.

Now that the network is being seen by the mobiles, we proceed to register the international mobile subscriber identity (IMSI) numbers of the SIM cards for establishing the mobile network. Use the following command in OpenBTS CLI to allow all the mobiles to connect to OpenBTS:

```
config Control.LUR.OpenRegistration .*
```

Now check the IMSI numbers of the connected mobiles using the following command:

```
OpenBTS > tmsis
IMSI TMSI IMEI AUTH CREATED ACCESSED TMSI ASSIGNED
214057715229963 - 012546629231850 0 78s 78s 0
001010000000002 - 312547515229963 1 80h 95s 0
001010000000003 - 351771053005400 1 80h 108s 0
```

Here, a '1' in the AUTH column indicates the LUR (Location Update Request) succeeded due to being a known subscriber and a '0' indicates LUR failed due to the mobile not being a known subscriber.

Step 9: Sending SMS Between Computer and Mobile

For enabling messaging between mobile phones we need to text the phone number of each mobile phone (here 2201001 and 2202002) to 101 from the respective phones for registering the mobile numbers.

```
Got SMS '273-lnobw' from IMSI312547515229963 for
    101.
Responding with "202 Queued".
Short-code SMS 101(2201001).
answering "Your phone is registered as .".
```

After registering the mobiles, send a text from one mobile to the other using their respective registered numbers, as seen in Fig. 3 and 4. In this case, the registered mobile numbers are 2201001 and 2202002.

Doing this will test that both the devices and the network are working properly before proceeding to the next step of making calls from one mobile phone to another. If both mobiles can receive texts from one another, then one can move onto the next step. However, if the texts either do not send or are not received, carefully check the configuration from Step 8 to make sure that everything is set up properly.

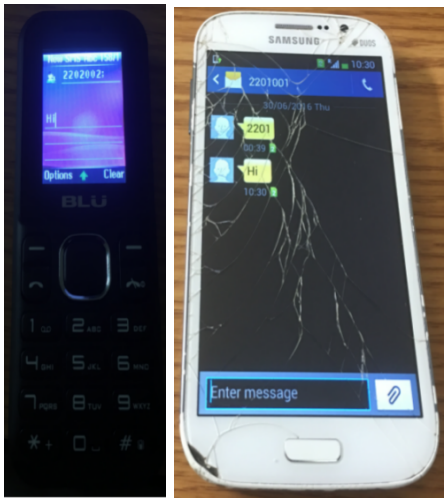

Fig. 3. Messaging from 2201001 to 2202002.

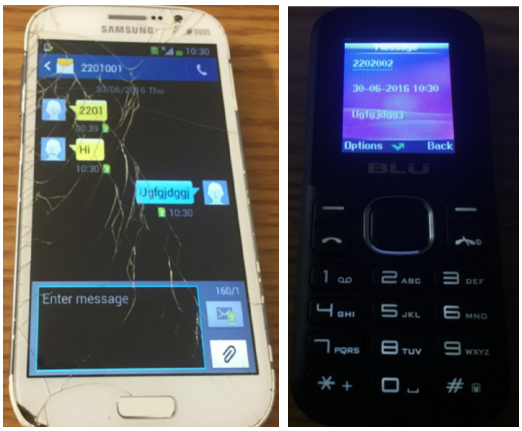

Fig. 4. Messaging from 2202002 to 2201001.

Step 10: Calling Between Mobiles

NOTE : We use the Asterisk software for calling between mobiles and configure the files (extensions.conf and sip.conf) accordingly to enable GSM calling functions between mobiles After configuring the Asterisk files (extennsions.conf, sip.conf) we will now be able to make calls between the registered mobiles located within the USRP-OpenBTS network.

/etc/asterisk/extensions.conf:

```
[macro-dialGSM]
exten => s,1,Dial(SIP/${ARG1},20)
exten => s,2,Goto(s-${DIALSTATUS},1)
exten => s-CANCEL,1,Hangup
exten => s-NOANSWER,1,Hangup
exten => s-BUSY,1,Busy(30)
exten => s-CONGESTION,1,Congestion(30)
exten => s-CHANUNAVAIL,1,playback(ss-noservice)
exten => s-CANCEL,1,Hangup
[sip-external]
exten => 2201001,1,Macro(dialGSM,IMSI310260424946902@127.0.0.1:5062)
exten => 2202002,1,Macro(dialGSM,IMSI310260419885747@127.0.0.1:5062)
```

/etc/asterisk/sip.conf:

```
[IMSI310260424946902]
callerid=2201001
canreinvite=no
type=friend
allow=gsm
context=sip-external
host=dynamic
dtmfmode=info

[IMSI310260419885747]
callerid=2202002
canreinvite=no
type=friend
allow=gsm
context=sip-external
host=dynamic
dtmfmode=info
```

Fig. 5. Configuring Asterisk.

Fig. 6. Calling from 2201001 to 2202002.

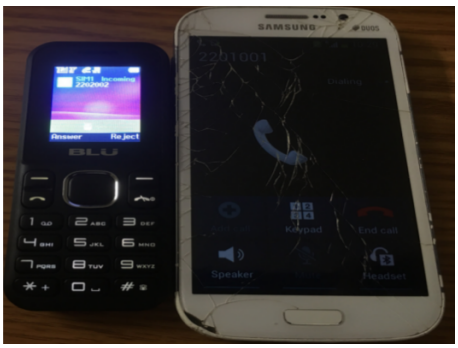

Fig. 7. Calling from 2202002 to 2201001.

IV. Results

A. Field Test:

TABLE II
PARAMETERS SET BEFORE ESTIMATING PERFORMANCE OF NETWORK.

Parameter	Value
Frequency of operation	900 MHz
Transmitted power of BTS	23dBm
BTS gain	3dB
Transmitted power of MS	33dBm
Height of antenna at BTS	24.2cm
Noise RSSI	-63dB
Uplink SNR cutoff	10dB

Before testing the performance of the network we fix the parameters as listed in table II. Testing is conducted in such a way that the host computer, USRP and one of the subscriber to the network (Mobile 2201001) are made stationary at a point while another subscriber (Mobile 2202002) moves to a certain distance intervals and initiates calls. The performance metrics like uplink signal to noise ratio (UPLINK SNR) in dB, uplink frame erasure rate (UPLINK FER), downlink received signal strength indicator (DOWNLINK RSSI) in dBm and downlink bit error rate (DOWNLINK BER) of mobile calls are analyzed and plotted against the intervals of distance. All these readings are calculated using “chans” command available in OpenBTS terminal and averaging is considered for consistency. Fig. 8 gives the Ariel view of testing site which is located at Oxford, Mississippi. The

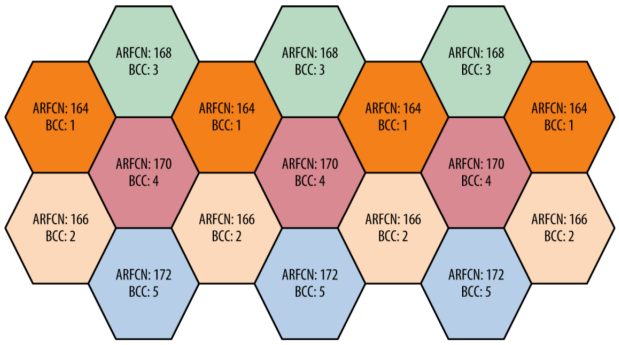

Fig. 13. MULTI-BTS.

V. Conclusion

The above implementation creates a complete, self-contained cellular network with minimal hardware (a computer and a USRP device). It is light, portable, cheap, and simple to install. Additionally, all software used is free and open-source. Thus, the implementation can find utility in applications including SAR missions in areas with compromised communication infrastructures, private networks for a business or work site, or research and development.

Acknowledgments

This work is supported by NASA EPSCoR program under grant NNX14AN38A.

References

[1] T. Rappaport, Wireless communications, 1st ed. Upper Saddle River, N.J.: Prentice Hall PTR, 2002.

[2] ”OpenBTS”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/OpenBTS. [Accessed: 23- Nov- 2016].

[3] ”GSM”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/GSM. [Accessed: 23- Nov- 2016].

[4] ”2G”, En.wikipedia.org, 2016. [Online]. Available: https://en.wikipedia.org/wiki/2G. [Accessed: 23- Nov- 2016].

[5] ”ECEN 4652/5002, Communications Lab, Spring 2016 - GNU Radio Installation”, Ecee.colorado.edu, 2016. [Online]. Available: http://ecee.colorado.edu/ mathys/ecen4652/SDRsoftware/GNURadioInstall.html. [Accessed: 23- Nov- 2016].

[6] ”Instalasi dan Konfigurasi OpenBTS 5.0”, My notebook, 2016. [Online]. Available: https://antonraharja.com/2016/03/16/instalasi-dan-konfigurasi-openbts-5-0/. [Accessed: 23- Nov- 2016].

[7] ”Asterisk 13 and FreePBX 13 on Ubuntu 14.04”, My notebook, 2016. [Online]. Available: https://antonraharja.com/2016/03/12/asterisk-13-and-freepbx-13-on-ubuntu-14-04/. [Accessed: 23- Nov- 2016].

[8] ”OpenBTS 5.0 install”, MessageLoop Footprints, 2016. [Online]. Available: https://xmsg.org/wordpress/?p=819. [Accessed: 23- Nov- 2016].

[9] ”Asterisk (PBX)”. En.wikipedia.org. N.p., 2017. Web. 24 Jan. 2017.

[10] Anthony, Faustine, Maria Gabriel, and Bertha Shao. ”Open Source Cellular Technologies for Cost Effective Cellular Connectivity in Rural Areas.” International Journal of Computer Applications 146, no. 15 (2016).