Installation Guide for OpenBTS

DOCUMENTATION

Christoph Kemetmüller

Center for Advanced Security Research Darmstadt

February 2010

Contents

Chapter 1

Overview

1.1 Introduction

This document provides an overview of the OpenBTS installation as part of the Global System for Mobile Communications (GSM) security research at the Center for Advanced Security Research Darmstadt (CASED). The hardware for the project is based on the Universal Software Radio Peripheral (USRP). We likewise did adapt OpenBTS to our needs while managing it in a virtual machine for obvious reasons, e.g., maintainability. The rest of this document is structured as follows.

The technical aspects and required hardware are described in Chapter 2. First of all, Section 2.1 gives an introduction to the USRP. The external clock used in this example realization is examined in Section 2.2. Section 2.3 summarizes up the modifications that had to be implemented in order to enable the clock input on the USRP. Finally, a list of all the utilized hardware equipment is presented in Section 2.4.

Chapter 3 provides a detailed guide on setting up a virtual machine with OpenBTS. The required software dependencies are listed in Section 3.1. Installing the essential GNU Radio software package is described in Section 3.2. The configuration and compilation of OpenBTS is described in Section 3.3. At the end of this Chapter in Section 3.4 stands a description for testing USRP connectivity and functionality.

Furthermore, the instructions and essential information for configuring and running OpenBTS are given in Chapter 4. An overview of the essential configuration options is presented in Section 4.1. Starting and operating OpenBTS is explained in Section 4.2. Problems encountered during the tests and field experiments are listed in Section 4.3.

The legal aspects of running a GSM network are described in Chapter 5. Section 5.1 presents details on acquiring a frequency allowance in Germany.

All scripts and patches mentioned in this document are appended at the end of this documentation.

Chapter 2

Hardware

2.1 Universal Software Radio Peripheral

The USRP from Ettus Research LLC is a software-defined radio (SDR) device and was chosen to be the basis for the realization [3]. The USRP is a USB-based motherboard for signal processing and offers open design as well as adaptability via its daughterboards. Each of these pluggable boards is equipped for a particular frequency range and transmission requirements.

For our implementation transmitting in the 1800 megahertz (MHz) frequency band was necessary. This frequencies are used for telecommunications in Germany and a test license for this particular frequency band was issued by the *Bundesnetzagentur* (see Chapter 5). Therefore, two RFX1800 daughterboards were required, one for transmitting and the other one for receiving. Each of these two boards was equipped with a VERT-900 antenna.

2.2 External Clock

During the first phase some problems with the USRP clock arose. The device was not able to maintain a certain level of clock accuracy required for GSM. Solution to this problem was finding an external clocking device with an appropriate accuracy that could replace the internal clock [6]. The *FA-SY 1* assembly kit by the German magazine for amateur radio and radio technology “funkamateur“ [4] was chosen to resolv this problem. This kit is able to provide the required clock accuracy with its on-board heating while coming for a acceptable price. Furthermore, it does also fit into the casing of the USRP (see Figure 2.2).

Essential aspect with this hardware device is configuring it to the correct output frequency of 64 MHz. The process of assessing the clock accuracy happened over a couple of days. During this time period the accuracy was measured by multiple frequency counters to monitor the behavior of the external clock. Finally, the device was was running with a drift of about 8

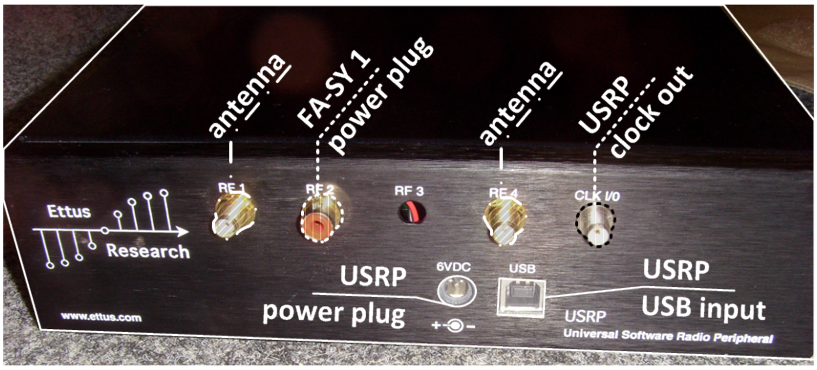

Figure 2.1: The back panel and the designated plugs as seen on our USRP.

Hz from the desired 64 MHz, which corresponds to a variation of 0.125 parts per million (ppm).

2.3 Hardware Modifications

In order to run the system with the external clocking device that is described in Section 2.2 some modifications to the USRP were necessary. The particular steps to disable the internal clock and enable the SubMiniature version A (SMA) connector for the clock input, as described in [9, 10], are:

1. Solder SMA connector to J2001
2. Move R2029 to R2030
3. Move C925 tot C926 (0.01uF)
4. Remove C924

After these modifications the *FA-SY 1* can be connected to the USRP via the SMA connector. Furthermore, the external clock was mounted into the USRP housing. The last step in preparing the hardware system was to wire up the systems in order to minimize the necessity of open the housing.

2.4 Equipment Summary

The hardware equipment consists of

- One USRP revision 4.5, S/N 4413 (see Figure 2.2)
  - Two RFX 1800 daughterboards
  - Two VERT-900 antennas

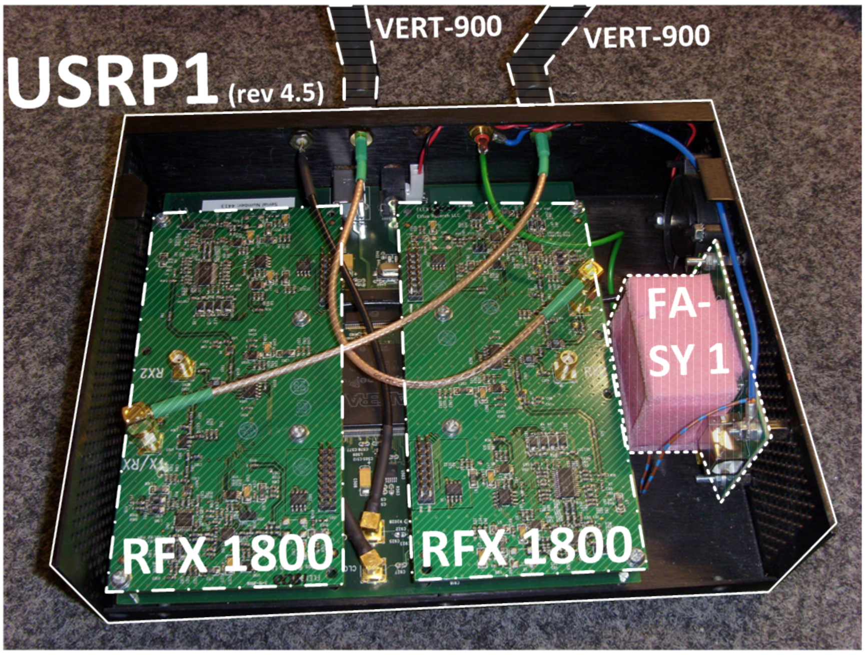

Figure 2.2: The inside of the USRP with the external clock build into the USRP housing.

  - One FA-SY 1

- One power adaptor (6V; 3.5A; US) + plug adaptor (DE) (see Figure 2.3)
- One power adaptor (12V; 2A) (see Figure 2.3)
- USB cable

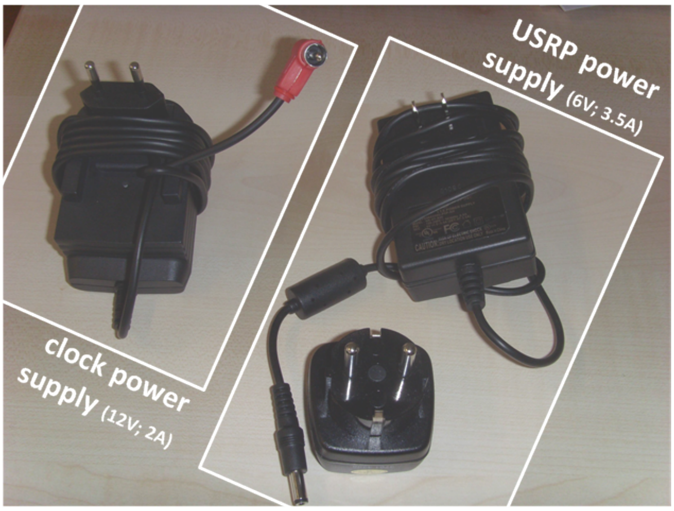

Figure 2.3: One 6V power adaptor for the USRP and the 12V adaptor for the external clock.

Chapter 3

Installation

This Chapter gives an overview of the steps required to install a working virtual machine. A general overview of the dependencies and steps required to install OpenBTS can be found under [2]. The particular steps executed during the installation of this realization are described hereafter. Furthermore, a complete summary of executed commands in form of a shell script is given in Listing C.1.

3.1 Preparing the Basic System

The operating system for running the example should be installed in a virtual machine called *OpenBTS* for the sake of better maintainability. Ubuntu 9.10 *Karmic Koala* was chosen as the operating system because of easy system maintenance. The OS setup was executed relying on the recommendations of the installer.

The first step after finishing the system setup was updating the software. Thereafter, the required software packages that could be added via the package management were installed on the system (see Listing 3.1).

```
sudo aptitude -y install swig automake1.9 libtool python-dev \
   libcppunit-dev sdcc libusb-dev libasound2-dev libsdl1.2-dev \
   patch python-wxgtk2.8 subversion guile-1.8-dev libqt4-dev \
   ccache python-opengl libgsl0-dev python-cheetah python-lxml \
   libqwt5-qt4-dev libqwtplot3d-qt4-dev qt4-dev-tools asterisk \
   fftw3-dev doxygen python-numpy-ext libosip2-dev libortp7-dev \
   git-core libssl-dev libwbxml2-utils patch git-core \
   build-essential
```

Listing 3.1: Packages required by GNU Radio and OpenBTS.

3.2 Installing GNU Radio

OpenBTS 2.5 depends on GNU Radio version 3.2.2 and later. One requirement that could not be installed via the package management was the particular version 1.37.0 of the boost library [1]. This library offers threading functionality that enhances the way GNU Radio operates.

```
if [ ! -f ${BOOSTGZ} ]; then
  _exec "wget http://kent.dl.sourceforge.net/sourceforge/boost/${
      BOOSTGZ}"
fi
if [ -d ${BOOST} ]; then
  _exec "rm -r ${BOOST}"
fi
_exec "tar -xf ${BOOSTGZ}"

if [ -d ${BOOST} ]; then
  _exec "cd ${BOOST}"
  _exec "./configure --with-libraries=thread,date_time,program_options
      "
  _exec "make"
  _exec "sudo make install"

  cd ..
fi

#required for gnuradio
if [ ! -f "/usr/local/include/boost" ]; then
  _exec "rm -r /usr/local/include/boost"
fi
_exec "ln -s /usr/local/include/boost-1_37/boost /usr/local/include/
    boost"
```

Listing 3.2: Installing the boost libraries.

The particular revision of GNU Radio used for the example is `11664`, which was available in the corresponding repository at http://gnuradio.org/svn/gnuradio/trunk/. As the GNU project was restructured in January 2010 and moved from subversion to git the source code is now available via http://gnuradio.org/git/gnuradio.git/. These changes have been incorporated into the installer script. The realization was successfully setup utilizing the tree-ish “e566be1bb983a0f4f284081760b6f91d9986d394“.

When trying to compile the source code right after the checkout a problem with finding Python.h occurred. The solution was to apply the patch in Listing B.1. After these changes GNU Radio compiled successfully and could be installed in the virtual machine.

```
GNURADIO="gnuradio-trunk"
if [ ! -d ${GNURADIO} ]; then
  _exec "git clone http://gnuradio.org/git/gnuradio.git ${GNURADIO}"
fi
if [ -d ${GNURADIO} ]; then
```

```
  _exec "cd ${GNURADIO}"

  _exec "sudo git clean -d -f"
  _exec "sudo git checkout e566be1bb983a0f4f284081760b6f91d9986d394 ."

  #patch gnuradio
  patch -N -b -p1 < ../gnuradio.bootstrap.patch
  patch -N -b -p1 < ../gnuradio.configure.ac.patch

  _exec "./bootstrap"
  _exec "./bootstrap" #bootstrap.patch wont work without
  _exec "./configure --disable-all-components --enable-usrp --enable-omnithread --enable-mblock --enable-pmt --enable-gruel"
  _exec "make && make check"
  _exec "sudo make install"

  cd ..
fi
```

Listing 3.3: Compiling gnuradio after applying the necessary patches.

In order to allow a non-root user access to the hardware device some other changes on the system were made. The group “usrp“ was added and was declared owner of the USRP by adding a corresponding rule to the udev configuration (see Listing 3.4).

```
sudo addgroup usrp
_exec "sudo usermod -G usrp -a ${USERNAME}"
echo 'ACTION=="add", BUS=="usb", SYSFS{idVendor}=="fffe", SYSFS{idProduct}=="0002", GROUP:="usrp", MODE:="0660"' > tmpfile
sudo chown root.root tmpfile
_exec "sudo mv tmpfile /etc/udev/rules.d/10-usrp.rules"

USRP_BYTESEX="/usr/local/include/usrp_bytesex.h"
USRP_STANDARD="/usr/local/include/usrp_standard.h"
USRP_PRIMS="/usr/local/include/usrp_prims.h"

if [ ! -f ${USRP_BYTESEX} ]; then
  _exec "rm ${USRP_BYTESEX}"
fi
if [ ! -f ${USRP_STANDARD} ]; then
  _exec "rm ${USRP_STANDARD}"
fi
if [ ! -f ${USRP_PRIMS} ]; then
  _exec "rm ${USRP_PRIMS}"
fi

_exec "sudo ln -sf /usr/local/include/usrp/usrp_bytesex.h ${USRP_BYTESEX}"
_exec "sudo ln -sf /usr/local/include/usrp/usrp_standard.h ${USRP_STANDARD}"
_exec "sudo ln -sf /usr/local/include/usrp/usrp_prims.h ${USRP_PRIMS}"
```

Listing 3.4: Setting up the udev permissions

3.3 Installing OpenBTS

One library required by OpenBTS is ortp, which provides the SIP functionality. The required version 0.13.1 available via [7] was missing some files. A later version [8] of ortp provided the necessary files and after copying them to the respective directory the library could be compiled and was installed successfully.

```
ORTP13="ortp-0.13.1"
ORTP13GZ="${ORTP13}.tar.gz"
ORTP16="ortp-0.16.1"
ORTP16GZ="${ORTP16}.tar.gz"

if [ ! -f ${ORTP13GZ} ]; then
  _exec "wget http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/${ORTP13GZ}"
fi
if [ ! -d ${ORTP13} ]; then
  tar -xf ${ORTP13GZ}
fi

if [ ! -f ${ORTP16GZ} ]; then
  _exec "wget http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/${ORTP16GZ}"
fi
if [ ! -d ${ORTP16} ]; then
  tar -xf ${ORTP16GZ}
fi

#tests aus ortp 0.16.1 nehmen
_exec "cp -r ${ORTP16}/src/tests ${ORTP13}/src/"

if [ -d ${ORTP13} ]; then
  cd ${ORTP13}
  make clean

  _exec "./configure"
  _exec "make"
  _exec "sudo make install"

  cd ..
fi
```

Listing 3.5: Installing the ortp libraries.

The steps executed so far were necessary as to install the required software in order to permit the installation of OpenBTS. OpenBTS 2.5.1 codename Lacassine was released in late 2009 and was used for this realization [5]. As gcc 4.4 does not automatically include the C++ standard libraries, e.g., `stdio.h`, `stdint.h`, #includes had to be added to some files of OpenBTS (see Listing B.4). Furthermore, to configure OpenBTS for the particular 1800 MHz frequency spectrum the particular constants in

`Transceiver/USRPDevice.h` had to be modified (see Listing B.3).

3.4 Testing the USRP

The final step after installing all the essential software dependencies is testing the functionality of the USRP. The SDR has to be setup by wiring up the device and connecting it to the virtual machine. As soon as the USRP is discovered by the *OpenBTS* VMware similar log messages as presented in Listing 3.6 are generated in `/var/log/messages`.

```
Jan 29 03:08:07 ubuntu kernel: [ 382.968052] usb 1-1: new high speed
     USB device using ehci_hcd and address 3
Jan 29 03:08:08 ubuntu kernel: [ 383.138699] usb 1-1: configuration
    #1 chosen from 1 choice
```

Listing 3.6: Excerpt from /var/log/messages that shows up upon connecting the USRP with the system.

Additionally, one can also use utilities for testing USRP functionality that come with the GNU Radio software package. With `usrp_benchmark_usb.py` (see Listing 3.7) the throughput of the USRP can be tested, which also implies connectivity of the SDR.

```
cd /usr/local/share/gnuradio/examples/usrp/
./usrp_benchmark_usb.py
```

Listing 3.7: Testing connectivity and throughput of the USRP.

Chapter 4

Running OpenBTS

4.1 Configuration

The configuration used during the tests is appended in Section C.2. Essential for running OpenBTS is choosing the appropriate Mobile Country Code (MCC) and Mobile Network Code (MNC). The most relevant identifiers for the tests are listed in Table 4.1. In the configuration file `OpenBTS.config` (see Listing C.2) these values are defined with `GSM.MNC` and `GSM.MCC`.

For running the system with the correct frequency two more adaptions were necessary; apart from to applying the patch in Listing B.3. The value for `GSM.Band` had to be changed to the desired frequency, `1800` for this realization. Furthermore, the correct Absolute Radio Frequency Channel Number (ARFCN) had to be chosen. This value determines the particular GSM channels to use. For the field test with this prototype the *Bundesnetzagentur* issued a test license for the ARFCN 866. This corresponds to an uplink frequency of 1781 MHz ($1710.2 + 0.2 * (n - 512)$) and a downlink frequency of 1876 MHz ($f_{up}(n) + 95$).[1]

[1] The frequencies can also be calculated using http://www.aubraux.com/design/arfcn-calculator.php (accessed 02/17/2010)

MCC	MNC	Country	Provider
001	01	-	TestSIM
262	01	DE	T-Mobile
262	02	DE	Vodafone
262	03	DE	E-Plus
262	07	DE	O2

Table 4.1: Provider identification codes

command	description
tmsis	Lists the associated IMSIs and their respective TMSIS.
tmsis clear	Delete all stored IMSIs.
sendsms <IMSI> <SRC>	Send a text message to the [IMSI] from number [SRC].
rolllac	Increment the cell's Location Area Code (LAC) by one.
cellid	Shows the current cell id information.
help	Lists all available commands.
help <cmd>	Get information about a particular command.
exit	Exit OpenBTS

Table 4.2: These are the most relevant commands available in the OpenBTS management console.

4.2 Operating OpenBTS

OpenBTS can be started with the executable `./apps/OpenBTS`. Sometimes problems might occur during the startup; these are further described in Section 4.3.

As soon as OpenBTS started successfully the software presents a management console to the user. The available commands can be listed by entering `help` and the most relevant ones for our tests are listed in Table 4.2

4.3 Potential Problems

As the whole system is very complex there are quite some sources for potential errors and problems. The ones we encountered during out tests are listed in the following paragraphs. Additional errors and hints on problems with running OpenBTS are listed in [2].

OpenBTS startup error

Sometimes the VMware image was successfully connected to the USRP, according to the logging messages in `/var/log/messages`, and yet OpenBTS would fail to start with logging messages similar to those presented in Listing 4.1. Unplugging the device and plugging it in again is one solution although not always solving the problem. Alternatively, rebooting the virtual machine is another option.

```
1266484661.4156 WARNING 140737354008336 TRXManager.cpp:271: sendCommandPacket: retrying transceiver command after response timeout
1266484662.4405 WARNING 140737354008336 TRXManager.cpp:271: sendCommandPacket: retrying transceiver command after response timeout
1266484663.4680 WARNING 140737354008336 TRXManager.cpp:271: sendCommandPacket: retrying transceiver command after response timeout
1266484663.4681 ERROR 140737354008336 TRXManager.cpp:287: sendCommandPacket: lost control link to transceiver
```

Listing 4.1: Error messages corresponding to a problem when starting the OpenBTS base station.

Stray transceiver process

As soon as OpenBTS has started successfully it dispatches a “transceiver“ process. This process can end up as *defunc*, thus preventing the program from starting again by blocking a system sockets.

Solution to this problem is manually killing the process by executing `sudo pkill -KILL transceiver`

Stuck Networkname

Some mobile phones would show the base station’s short name as defined with `GSM.ShortName` in the configuraiotn file even after rebooting the handset. One solution to this problem is trying to cold boot the device, which requires removing the battery from the mobile phone. Another option is to manually change the `GSM.ShortName` value to the desired text and restarting the base station; this works especially well to calm down furious victims of a field experiment.

Chapter 5

Legal Aspects

5.1 Radio Regulations

Running radio equipment and working with a GSM base station requires transmitting in the officially regulated radio spectrum. Therefore, to legally run a base station an official allowance for a particular frequency is necessary. Here in Germany the *Bundesnetzagentur* issues these licenses for limited time and for a specific location. A testing license is cost free for German universities in order to encourage research and education in this area.

Appendix A

Relevant Sources

Apart from the sources listed in the bibliography, this Chapter represents a summary of essential and important links for additional information and furher reading material.

Hardware

- General information
- USRP User's and Developer's Guide
- Kestrel OpenBTS Store
- European merchant
- Connect external clock to USRP
- FA-Synthesizer (FA-SY 1)

OpenBTS

- OpenBTS Project homepage
- OpenBTS Discussion board
- The OpenBTS Wiki Subspace
- The OpenBTS Chronicles (blog)
- Installing OpenBTS and GnuRadio with Fedora Core 11

OMA Client Provisioning

- OMA Client Provisioning
- Provisioning Content
- PDU format
- Settings provisioning to mobile devices
- WSP header
- WAP Technical Discussions (unsupported)
- SMS over 3GPP IMS Network

Related Projects

- OpenBSC
- The OpenBSC Archives
- AirProbe

Research

- Hijacking Mobile Data Connections
- 26C3: GSM SRSLY?
- 26C3: Using OpenBSC for fuzzing of GSM handsets
- 26C3: Playing with the GSM RF Interface

Other

- GNU Radio Wiki
- ARFCN

Persons

- Dipl.-Inf. Michael Müller (Electronic egnineer)
- Prof.Dr. Michael Massoth (Professor in telecommunications)
- David Burgess
- Harald Welte

Appendix B

Modifications

This Chapter includes all the patches and software modifications created during this project.

The patch files are color-formatted in order to quickly get an overview of the changes. Red represents deletions from the source code, while green lines indicate new code. Orange lines are just informational and are limited to the line and row information of the patches.

```
--- gnuradio-trunk/configure.ac (revision 11664)
+++ gnuradio-trunk/configure.ac (working copy)
@@ -125,6 +125,7 @@
 AC_DISABLE_STATIC dnl don't build static libraries
 m4_ifdef([LT_INIT],[LT_INIT],[AC_PROG_LIBTOOL])
 GR_FORTRAN
+AC_PROG_CC

 GR_NO_UNDEFINED  dnl do we need the -no-undefined linker flag
 GR_SCRIPTING
```

Listing B.1: Patch for the gnuradio autoconfigure script in order to solve an error finding textitPython.h.

```
--- a/bootstrap
+++ b/bootstrap
@@ -23,11 +23,11 @@
 rm -fr config.cache autom4te*.cache

 aclocal  -I config
-autoconf
-autoheader
  libtoolize  --automake
 automake --add-missing -Wno-portability -Wno-override -Wnone
 #automake --add-missing -Wno-portability
+autoconf
+autoheader

```

```
# Run bootstrap in any subprojects
(cd usrp2/firmware ; ./bootstrap)
```

Listing B.2: Patch for gnuradio boostrap script to solve an issue with a missing *Makefile.in*.

```
--- openbts-2.5Lacassine/Transceiver/USRPDevice.h 2009-11-01 16:00:30.000000000 -0800
+++ openbts-2.5/Transceiver/USRPDevice.h 2009-12-01 08:34:39.102135221 -0800
@@ -119,11 +119,11 @@
   static const unsigned CP2 = 7;
   static const unsigned CP1 = 7;
   static const unsigned DIVSEL = 0;
-  static const unsigned DIV2 = 1;
-  static const unsigned freq_mult = 2;
+  static const unsigned DIV2 = 0;
+  static const unsigned freq_mult = 1;
   static const unsigned CPGAIN = 0;
-  static const float    minFreq = 800e6;
-  static const float    maxFreq = 1000e6;
+  static const float    minFreq = 1600e6;
+  static const float    maxFreq = 2000e6;
   static const float    freqRes = 4e6;

   // R-Register Common Values
```

Listing B.3: OpenBTS is configured to transmit in the 1800 MHz frequency range with these changes.

```
--- openbts-2.5Lacassine/CommonLibs/F16.h 2009-10-25 20:07:03.000000000 -0700
+++ openbts-2.5/CommonLibs/F16.h 2009-12-01 07:27:04.861413793 -0800
@@ -26,7 +26,7 @@
 #ifndef F16_H
 #define F16_H

-
+#include <stdint.h>
 #include <ostream>


--- openbts-2.5Lacassine/CommonLibs/Logger.h 2009-10-12 22:15:38.000000000 -0700
+++ openbts-2.5/CommonLibs/Logger.h 2009-12-01 08:28:35.051252031 -0800
@@ -27,6 +27,7 @@
 #define LOGGER_H

 #include <sstream>
```

```
+#include <stdio.h>
 #include "Threads.h"


--- openbts-2.5Lacassine/Transceiver/USRPping.cpp 2009-10-21 15:15:35.000000000 -0700
+++ openbts-2.5/Transceiver/USRPping.cpp 2009-12-01 08:26:26.001827733 -0800
@@ -24,6 +24,7 @@



+#include <stdio.h>
 #include "Transceiver.h"
 #include <Logger.h>
```

Listing B.4: Changes required when compiling OpenBTS 2.5.1 Lacassine with gcc 4.4.

Appendix C

System Configuration

The installation script for automatic setup of a basic OpenBTS system is appended in this Chapter. Additionally, the configuration file for OpenBTS that was used during the tests is also present in this part of the document.

```
USERNAME=‘whoami‘
STAGE=0

function _exec {
  cmd=$1
  echo $cmd
  eval ${cmd}
  ret=$?

  if [ ${ret} -ne 0 ]; then
    echo "command ’${cmd}’ not execute successfully (${ret})"
    exit
  fi
}

function aptinstalls {
  _exec "sudo aptitude update"
  _exec "sudo aptitude -y install swig automake1.9 libtool python-dev \
     libcppunit-dev sdcc libusb-dev libasound2-dev libsdl1.2-dev \
     patch python-wxgtk2.8 subversion guile-1.8-dev libqt4-dev \
     ccache python-opengl libgsl0-dev python-cheetah python-lxml \
     libqwt5-qt4-dev libqwtplot3d-qt4-dev qt4-dev-tools asterisk \
     fftw3-dev doxygen python-numpy-ext libosip2-dev libortp7-dev \
     git-core libssl-dev libwbxml2-utils patch git-core \
     build-essential"
}

#boost 1.37.0
BOOST="boost_1_37_0"
BOOSTGZ="${BOOST}.tar.gz"
BOOST_PREFIX="/opt/boost_1_37_0"

function boost {
  if [ ! -f ${BOOSTGZ} ]; then
```

```
    _exec "wget http://kent.dl.sourceforge.net/sourceforge/boost/${BOOSTGZ}"
  fi
  if [ -d ${BOOST} ]; then
    _exec "rm -r ${BOOST}"
  fi
  _exec "tar -xf ${BOOSTGZ}"

  if [ -d ${BOOST} ]; then
    _exec "cd ${BOOST}"
    _exec "./configure --with-libraries=thread,date_time,program_options"
    _exec "make"
    _exec "sudo make install"

    cd ..
  fi

  #required for gnuradio
  if [ ! -f "/usr/local/include/boost" ]; then
    _exec "rm -r /usr/local/include/boost"
  fi
  _exec "ln -s /usr/local/include/boost-1_37/boost /usr/local/include/boost"
}

#gnuradio
function gnuradio {
#svn -r 11664 co http://gnuradio.org/svn/gnuradio/trunk/ gnuradio-trunk
  GNURADIO="gnuradio-trunk"
  if [ ! -d ${GNURADIO} ]; then
    _exec "git clone http://gnuradio.org/git/gnuradio.git ${GNURADIO}"
  fi
  if [ -d ${GNURADIO} ]; then
    _exec "cd ${GNURADIO}"

    _exec "sudo git clean -d -f"
    _exec "sudo git checkout e566be1bb983a0f4f284081760b6f91d9986d394 ."

    #patch gnuradio
    patch -N -b -p1 < ../gnuradio.bootstrap.patch
    patch -N -b -p1 < ../gnuradio.configure.ac.patch

    _exec "./bootstrap"
    _exec "./bootstrap" #bootstrap.patch wont work without
    _exec "./configure --disable-all-components --enable-usrp --enable-omnithread --enable-mblock --enable-pmt --enable-gruel"
    _exec "make && make check"
    _exec "sudo make install"

    cd ..
  fi

  #Allow non-root access to the USRP
  sudo addgroup usrp
  _exec "sudo usermod -G usrp -a ${USERNAME}"
  echo ’ACTION=="add", BUS=="usb", SYSFS{idVendor}=="fffe", SYSFS{idProduct}=="0002", GROUP:="usrp", MODE:="0660"’ > tmpfile
  sudo chown root.root tmpfile
  _exec "sudo mv tmpfile /etc/udev/rules.d/10-usrp.rules"

  USRP_BYTESEX="/usr/local/include/usrp_bytesex.h"
  USRP_STANDARD="/usr/local/include/usrp_standard.h"
  USRP_PRIMS="/usr/local/include/usrp_prims.h"

  if [ ! -f ${USRP_BYTESEX} ]; then
    _exec "rm ${USRP_BYTESEX}"
  fi
  if [ ! -f ${USRP_STANDARD} ]; then
    _exec "rm ${USRP_STANDARD}"
  fi
  if [ ! -f ${USRP_PRIMS} ]; then
    _exec "rm ${USRP_PRIMS}"
  fi

  _exec "sudo ln -sf /usr/local/include/usrp/usrp_bytesex.h ${USRP_BYTESEX}"
  _exec "sudo ln -sf /usr/local/include/usrp/usrp_standard.h ${USRP_STANDARD}"
  _exec "sudo ln -sf /usr/local/include/usrp/usrp_prims.h ${USRP_PRIMS}"

}

#ortp 0.13.1
function ortp {
  ORTP13="ortp-0.13.1"
  ORTP13GZ="${ORTP13}.tar.gz"
  ORTP16="ortp-0.16.1"
  ORTP16GZ="${ORTP16}.tar.gz"

  if [ ! -f ${ORTP13GZ} ]; then
    _exec "wget http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/${ORTP13GZ}"
  fi
  if [ ! -d ${ORTP13} ]; then
    tar -xf ${ORTP13GZ}
  fi

  if [ ! -f ${ORTP16GZ} ]; then
    _exec "wget http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/${ORTP16GZ}"
  fi
  if [ ! -d ${ORTP16} ]; then
    tar -xf ${ORTP16GZ}
  fi

  #tests aus ortp 0.16.1 nehmen
  _exec "cp -r ${ORTP16}/src/tests ${ORTP13}/src/"

  if [ -d ${ORTP13} ]; then
    cd ${ORTP13}
    make clean

    _exec "./configure"
    _exec "make"
    _exec "sudo make install"

    cd ..
  fi
}

#openBTS
OPENBTS="openbts-2.5Lacassine"
OPENBTS="openbts-2.5.1Lacassine"
OPENBTSGZ="${OPENBTS}.tar.gz"

function openbts {
  if [ ! -f ${OPENBTSGZ} ]; then
    _exec "wget http://downloads.sourceforge.net/project/openbts/${OPENBTSGZ}?use_mirror=dfn"
  fi
  if [ ! -d ${OPENBTS} ]; then
    tar -xf ${OPENBTSGZ}

    cd ${OPENBTS}
    patch -N -b -p1 < ../${OPENBTS}.USRPDevice.h.patch
    patch -N -b -p1 < ../${OPENBTS}.patch
    cd ..
  fi

  if [ -d ${OPENBTS} ]; then
    cd ${OPENBTS}
    #make clean

    #./bootstrap
    _exec "./configure"
    _exec "make"

    cd ..

    _exec "cp ${OPENBTS}.OpenBTS.config ./${OPENBTS}/apps/"
  fi
}



CHOICE=$1
echo "-- ${CHOICE} --"

if [ ! -z ${CHOICE} ]; then
  if [[ ${CHOICE} -le 1 ]]; then
    aptinstalls
  fi
  if [[ ${CHOICE} -le 2 ]]; then
    boost
  fi
  if [[ ${CHOICE} -le 3 ]]; then
    gnuradio
  fi
  if [[ ${CHOICE} -le 4 ]]; then
    ortp
  fi
  if [[ ${CHOICE} -le 5 ]]; then
    openbts
  fi
else
  STAGE=$((${STAGE}+1))
  echo "STAGE ${STAGE}"
  aptinstalls
  STAGE=$((${STAGE}+1))
  echo "STAGE ${STAGE}"
  boost
  STAGE=$((${STAGE}+1))
  echo "STAGE ${STAGE}"
  gnuradio
  STAGE=$((${STAGE}+1))
  echo "STAGE ${STAGE}"
  ortp
  STAGE=$((${STAGE}+1))
  echo "STAGE ${STAGE}"
  openbts
fi
```

Listing C.1: This script resembles the executed command during the installation of OpenBTS in the Ubuntu 8.10 virtual machine.

```
# Sample OpenBTS configuration file.
# Format of each line is. <key><space><value>
# The key name can contain no spaces.
# Everything between the first space and the end of the line becomes the value.
# Comments must start with "." at the beginning of the line.
# Blank lines are OK.

# As a gerenal rule, non-valid configuration values will crash OpenBTS.

#
# Logging parameters
#

# The initial global logging level: ERROR, WARNING, NOTICE, INFO, DEBUG, DEEPDEBUG
LogLevel INFO

# The log file path. If not set, logging goes to stdout.
LogFileName test.out

```

```
# Wireshark support
# The standard IANA for GSMTAP is 4729
# If if this is not defined, we do not generate the GSMTAP dumps.
Wireshark.Port 4729

# Port number for test calls.
# This is where an external program can interact with a handset via UDP.
TestCall.Port 28670

#
# Transceiver parameters
#

# Transceiver interface
# This TRX.IP is not really adjustable. Just leave it as 127.0.0.1.
TRX.IP 127.0.0.1
# This value is hard-coded in the transcevier. Just leave it alone.
TRX.Port 5700

# Path to transceiver binary
# YOU MUST HAVE A MATCHING libusrp AS WELL!!
TRX.Path ../Transceiver/transceiver

# TRX logging.
# Logging level.
TRX.LogLevel WARNING
# Logging file. If not defined, logs to stdout.
TRX.LogFileName test.TRX.out

#
# SIP, RTP, servers
#

# Asterisk PBX
#Asterisk.IP 192.168.0.15
Asterisk.IP 127.0.0.1
Asterisk.Port 5060

# Messaging server
Messenger.IP 127.0.0.1
Messenger.Port 5063

# Local SIP/RTP ports
SIP.Port 5062
RTP.Start 16484
RTP.Range 98

# If Asterisk is 127.0.0.1, this is also 127.0.0.1.
# Otherwise, this should be the local IP address of the interface used to contact asterisk.
SIP.IP 127.0.0.1
# This is broken; use localhost. SIP.IP 192.168.0.16

# Local SMS port for short code delivery.
```

```
SMSLoopback.Port 5064

#
# Special extensions.
#

# Routing extension for emergency calls.
PBX.Emergency 2101

#
# SIP parameters
#

# SIP registration period in seconds.
# Ideally, this should be slightly longer than GSM.T3212.
SIP.RegistrationPeriod 3600

#
# SIP Internal Timers. All timer values are given in millseconds.
# These are from RFC-3261 Table A.
#

# SIP Timer A, the INVITE retry period, RFC-3261 Section 17.1.1.2
SIP.Timer.A 1000

#
# SMS parameters
#
# ISDN address of source SMSC when we fake out a source SMSC.
SMS.FakeSrcSMSC 0000
# ISDN address of destination SMSC when a fake value is needed.
SMS.DefaultDestSMSC 0000

# The SMS HTTP gateway.
# Comment out if you don't have one or if you want to use smqueue.
#SMS.HTTP.Gateway api.clickatell.com

# IF SMS.HTTP.Gateway IS DEFINED, SMS.HTTP.AccessString MUST ALSO BE DEFINED.
SMS.HTTP.AccessString sendmsg?user=xxxx&password=xxxx&api_id=xxxx

# Open Registration and Self-Provisioning
# This is a bool and if set to 1, OpenBTS will allow all handsets to register
Control.OpenRegistration 1

#
# "Welcome" messages sent during IMSI attach attempts.
# ANY WELCOME MESSAGE MUST BE LESS THAN 161 CHARACTERS.
# ANY DEFINED WELCOME MESSAGE MUST ALSO HAVE A DEFINED SHORT CODE.
# Comment out any message you don't want to use.
#

```

```
# The message sent upon full successful registration, all the way through the Asterisk server.
Control.NormalRegistrationWelcomeMessage TestSIM network.
Control.NormalRegistrationWelcomeShortCode 0000

# Then message sent to accpeted open registrations.
# IF OPEN REGISTRATION IS ENABLED, THIS MUST ALSO BE DEFINED.
Control.OpenRegistrationWelcomeMessage Herzlich willkommen im CASED GSM-Netz.
Control.OpenRegistrationWelcomeShortCode 23

# Then message send to failed registrations.
Control.FailedRegistrationWelcomeMessage FAIL!
Control.FailedRegistrationWelcomeShortCode 666

#Control.MyWelcomeMessage Sie erhalten gleich ein sicherheitskritisches Systemupdate. Installieren Sie dieses oder schalten Sie ihr Mobiltelefon ab und kontaktieren den Kundenservice.
Control.MyWelcomeMessage Sicherheitsupdate! Installieren Sie dieses oder schalten Sie ihr Mobiltelefon ab und kontaktieren den Kundenservice.
Control.MyWelcomeShortCode 555

#
# GSM
#

# Network and cell identity.

# Network Color Code, 0-7
GSM.NCC 0
# Basesation Color Code, 0-7
GSM.BCC 0
# Mobile Country Code, 3 digits.
# US is 310
# MCC MUST BE 3 DIGITS. Prefix with 0s if needed.

# Test code is 001.
GSM.MCC 001
#Germany
#GSM.MCC 262

# Mobile Network Code, 2 or 3 digits.
# Test code is 01.

#T-Mobile
GSM.MNC 01
#vodafone
#GSM.MNC 02
#E-Plus
#GSM.MNC 03
#O2
#GSM.MNC 07

```

```
# Location Area Code, 0-65535
GSM.LAC 841
# Cell ID, 0-65535
GSM.CI 22666
# Network "short name" to display on the handset.
# SHORT NAME MUST BE LESS THAN 8 CHARACTERS.
GSM.ShortName vodafone

# Assignment type for call setup.
# This is defined in an enum AssignmentType in GSMCommon.h.
# 0=Early, 1=VeryEarly.
GSM.AssignmentType 1

# Band and Frequency

# Valid band values are 850, 900, 1800, 1900.
#GSM.Band 900
GSM.Band 1800

# Valid ARFCN range depends on the band.
#GSM.ARFCN 29
# ARCN 975 is inside the US ISM-900 band and also in the GSM900 band.
#GSM.ARFCN 975
# ARFCN 207 was what we ran at BM2008, I think, in the GSM850 band.
GSM.ARFCN 866

# Neightbor list
GSM.Neighbors 207

# Downlink tx power level, dB wrt full power
GSM.PowerAttenDB 0

# Channel configuration
# Number of C-VII slots (8xSDCCH)
GSM.NumC7s 1
# Number of C-I slots (1xTCH/F)
GSM.NumC1s 5

# Beacon parameters.

# L1 radio link timeout advertised on BCCH.
# This is the RAW parameter sent on the BCCH.
# See GSM 10.5.2.3 for encoding.
# Value of 15 gives 64-frame timeout, about 30 seconds on the TCH.
# This should be coordinated with T3109.
GSM.RADIO_LINK_TIMEOUT 15

# Control Channel Description (CCD)

# Attach/detach flag.
# Set to 1 to use attach/detach procedure, 0 otherwise.
# This will make initial registration more prompt.
# It will also cause an un-regstration if the handset powers off.
GSM.CCD.ATT 1
```

```

# CCCH.CONF
# See GSM 10.5.2.11 for encoding.
# Value of 1 means we are using a C-V beacon.
GSM.CCD.CCCH_CONF 1

# RACH Parameters

# Maximum RACH retransmission attempts
# This is the RAW parameter sent on the BCCH.
# See GSM 04.08 10.5.2.29 for encoding.
GSM.RACH.MaxRetrans 3

# Parameter to spread RACH busts over time.
# This is the RAW parameter sent on the BCCH.
# See GSM 04.08 10.5.2.29 for encoding.
GSM.RACH.TxInteger 14

# Access class flags.
# This is the RAW parameter sent on the BCCH.
# See GSM 04.08 10.5.2.29 for encoding.
# Set to 0 to allow full access.
GSM.RACH.AC 0

GSM.RACH.CellBarAccess 0

# NCCs Permitted.
# An 8-bit mask of allowed NCCs.
# Unless you are coordinating with another carrier,
# this should probably just select your own NCC.
GSM.NCCsPermitted 1

# Cell Selection Parameters (CS)

GSM.CS.MS_TXPWR_MAX_CCH 0
GSM.CS.RXLEV_ACCESS_MIN 0

# Cell Reselection Hysteresis
# See GSM 04.08 10.5.2.4, Table 10.5.23 for encoding.
# Encoding is 2N dB, value values of N are 0..7 for 0..14 dB.
GSM.CS.CELL_RESELECT_HYSTERESIS 7

# Reject cause for location updating failures
# Reject causes come from GSM 04.08 10.5.3.6
# Reject cause 0x04, IMSI not in VLR
GSM.LURejectCause 0x04

# Maximum TA for accepted bursts.
# Can be used to control the range of the BTS.
# The unit is GSM symbols of round trips delay, about 550 meters per symbol.
GSM.MaxRACHDelay 20

#
# GSM Timers. All timer values are given in milliseconds unless stated otherwise.
# These come from GSM 04.08 11.2.
#

# T3212, registration timer.
# Unlike most timers, this is given in MINUTES.
# Actual period will be rounded down to a multiple of 6 minutes.
# Any value below 6 minutes disables periodic registration, which is probably a bad idea.
# Valid range is 6..1530.
# Ideally, this should be slightly less than the SIP.RegistrationPeriod.
GSM.T3212 6

# T3122, RACH holdoff timer.
# This value can vary internally between the min and max ends of the range.
# When congestion occurs, T3122 grows exponentially.
GSM.T3122Min 2000
# T3211Max MUST BE NO MORE THAN 255 ms.
GSM.T3122Max 255000
```

Listing C.2: The configuration file as utilized during the tests with this example implementation.

Bibliography

[1] Boost C++ Libraries. Online http://www.boost.org (sighted 02/16/2010), s.d.

[2] Building and Running OpenBTS. Online http://gnuradio.org/redmine/wiki/1/OpenBTSBuildingAndRunning (sighted 02/22/2010), s.d.

[3] Ettus Research LLC | Order. Online http://www.ettus.com/order (sighted 02/16/2010), s.d.

[4] FA-SY 1. Online http://www.box73.de/catalog/product_info.php?products_id=1869&osCsid=f7t0mheibgj27p8h5oeknkdj54 (sighted 02/16/2010), s.d.

[5] OpenBTS 2.5.1 Lacassine. Online http://downloads.sourceforge.net/project/openbts/openbts-2.5.1Lacassine.tar.gz?use_mirror=switch (sighted 02/16/2010), s.d.

[6] OpenBTS/Clocks – openbts. Online http://sourceforge.net/apps/trac/openbts/wiki/OpenBTS/Clocks (sighted 02/25/2010), s.d.

[7] ortp 0.13.1. Online http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/ortp-0.13.1.tar.gz (sighted 02/16/2010), s.d.

[8] ortp 0.16.1. Online http://mirrors.zerg.biz/nongnu/linphone/ortp/sources/ortp-0.16.1.tar.gz (sighted 02/16/2010), s.d.

[9] Maritz, H. OpenBTS fix: Connect external clock to USRP. Online http://students.ee.sun.ac.za/~gshmaritz/?p=152 (sighted 02/16/2010), 2009.

[10] Radio, G. GNU Radio - UsrpFAQClocking - gnuradio.org. Online http://www.gnuradio.org/trac/wiki/USRPClockingNotes (sighted 12/22/2009), s.d.