Home

Jump to bottom

Sébastien Dudek edited this page on Jan 9 · 8 revisions

Welcome to the OpenBTS-UMTS wiki!

This wiki is a backup of the original one on http://openbts.org/w/index.php/OpenBTS-UMTS, but not available anymore for now.

Introduction

OpenBTS-UMTS is a Linux-based application that uses a software radio to present a UMTS network to any standard 3G UMTS handset or modem. It builds upon the OpenBTS framework, where the MS or UE is treated as an IP endpoint at the edge of the network.

Release Features/Capabilities

- supports original UMTS Release 99 (or Release 3)
- supports packet-switched services only (i.e. data)
- supports a single U-ARFCN
- supports one or two high-speed active data sessions
- spreading factors of 4-256
- rate-1/2 convolutional coding
- rate-1/3 turbo coding
- maximum downlink data speed of 106 Kbytes/s
- maximum uplink data speed of 52 Kbytes/s
- Integrity Protection of GSM SIMs
- Features not supported in this release:
  - circuit-switched services (e.g. voice, text)
  - handover
  - Inter-RAT mobility (moving b/w a 2G and 3G network)
  - paging
  - ciphering

- USIM-based authentication

Supported Hardware

Initial integration of the OpenBTS-UMTS public release was accomplished with the Range Networks SDR1 (a.k.a. RAD1). Support for recent Ettus Research USRP devices is now available. Integration with other software-defined radios is ongoing, see Radio Integration for more information about connecting OpenBTS and OpenBTS-UMTS to different radio interfaces.

Range Networks SDR1

The Range Networks SDR1 was designed primarily for 2G and 2.5G operation. Thus, operation with the OpenBTS-UMTS release can not be guaranteed with all SDR1 radios, since the USB2.0 interface is highly stressed to maintain the required data rates for 3G. Moreover, it requires a reprogramming of the SDR1's clock chip from 52Mhz to 61.44Mhz, which is described later on this page.

Ettus Research USRP

Supported Ettus Research products include third generation USRP devices (B200 series and X-series) and second generation models with capable bandwidth for UMTS (N-series). Older Ettus Research USB 2.0 based products are not supported by OpenBTS-UMTS due to transport bus limitations. Supported USRP devices include Intel SSE optimization for UMTS pulse shaping and host resampling operations.

Ettus Research UMTS Capable Devices

	Transport	Recommended RF	Frequency Accuracy
B200/B210/B205	USB 3.0	Integrated	TCXO 2.0 ppm
X300/X310	1 or 10 Gigabit Ethernet/PCI-Express	SBX, WBX, CBX	TCXO 2.5 ppm
N200/N210	1 Gigabit Ethernet	SBX, WBX, CBX	TCXO 2.5 ppm
USRP2	1 Gigabit Ethernet	SBX, WBX, CBX	VCXO 20 ppm

Not that clock accuracy can be optimized to <1 ppb with a GPSDO.

UMTS Band Support

	Frequency Range	UMTS Bands	Output Power
B200/B210/B205	70 MHz - 6 GHz	1-14, 19-21, 22, 25, 26, 32	up to 100 mW
WBX	50 Mhz - 2.2 GHz	1-14, 19-21, 25, 26, 32	TCXO 2.5 ppm
SBX	400 MHz - 4.4 GHz	1-14, 19-21, 22, 25, 26, 32	TCXO 2.5 ppm
CBX	1.2 GHz - 6 GHz	1-4, 7, 9-11, 21, 22, 25	VCXO 20 ppm

CPU requirements

OpenBTS-UMTS is a more computationally intensive application than OpenBTS, since the UMTS channel bandwidth is roughly 13x larger than a GSM channel. Generally, a multi-core high performance CPU is required, such as Intel Core i3, i5, or i7 running at more than 1.6Ghz. Intel Atom processors are too weak to support the current implementation.

Range Networks Development Kits

There's good news and bad news for users or Range Networks development kits. The good news is that you already have an SDR1 that may work. The bad news is that it is connected to an Atom-based CPU board, which is not suitable for UMTS. You will need to connect your SDR1 to a more powerful CPU board.

Phones/Modems tested

- Works
  - iPhones (3, 4, and 5)
  - HTC Velocity
  - Samsung Galaxy
  - Palm Pre
  - a variety of Multitech modems *Doesn't work
  - None so far. Please update at will.

SIMS and Authentication

UMTS mandates mutual authentication between the UE and the NodeB. This is a major change from 2G/2.5G authentication, where only the BTS authenticates the MS.

A good presentation of UMTS security is available at
http://www.netlab.tkk.fi/opetus/s38153/k2003/Lectures/q42UMTS_security.pdf

Another detailed description is available at http://www.3g4g.co.uk/Tutorial/ZG/zg_security.html

So what does this mean? The subscriber registry will need to know the SIM's K_i value to

perform authentication and enable integrity protection. Without proper authentication and integrity protection, the UE will not attach (or register) with OpenBTS-UMTS. For most users, this means you must provide the SIMs for the UEs on the network. The only way to use SIMs from another provider is to obtain the K_i through a roaming interface to the provider's HLR/HSS.

This also means that some of the features that circumvented authentication in OpenBTS, like open registration, are not possible with OpenBTS-UMTS.

USIMs (e.g. 3G SIMs) are not currently supported by the OpenBTS-UMTS implementation. They require different authentication algorithms than GSM SIMs; these algorithms are not supported in the public release of OpenBTS-UMTS.

Build, Install, Setup, and Run Instructions

The following prerequisites are required to build and run OpenBTS-UMTS:

- Development and build packages for your Linux distribution. These are mostly unchanged from OpenBTS. ASN1C compiler. Version 0.9.23 is required (provided in the repo).
- Sipauthserve for handset registration and processing handsets. See Installing the Subscriber Registry and Sipauthserve.
- Valid SIM with a known IMSI and Ki values. These values must be added to the subscriber registry

Obtaining the (unofficial) Source

OpenBTS-UMTS is available on GitHub, ensure that you have Git installed on your system.

```
git clone https://github.com/PentHertz/OpenBTS-UMTS.git
# Optionally use 1.1 branch for a stable version
```

The NodeManager component is setup as a git submodule and can be added with the following commands. Note that the current configuration requires a GitHub account to check out the submodule.

```
$ cd  OpenBTS-UMTS
$ git submodule init
$ git submodule update
```

Configure and Build

You are now ready to build OpenBTS-UMTS, from your checked-out OpenBTS-UMTS directory:

```
./install_dependences.sh # install all dependencies without caring
./autogen.sh
./configure
make
sudo make install
```

Ettus Research USRP

Make sure that the Ettus UHD driver was found in the output when running configure. When found, Intel SSE support will also be tested and automatically enabled. If UHD is not found, verify that the UHD driver is properly installed.

```
checking for UHD... yes
checking whether mmx is supported... yes
checking whether sse is supported... yes
checking whether sse2 is supported... yes
checking whether sse3 is supported... yes
checking whether ssse3 is supported... yes
checking whether sse4.1 is supported... yes
checking whether sse4.2 is supported... yes
```

The UHD device should be detectable before running OpenBTS. Device presence can be checked with the uhd_usrp_probe command installed with the UHD driver.

```
uhd_usrp_probe
[INFO] [UHD] linux; GNU C++ version 11.2.0; Boost_107400; UHD_4.1.0.5-3
[INFO] [B200] Loading firmware image: /usr/share/uhd/images/usrp_b200_fw.hex...
[INFO] [B200] Detected Device: B205mini
[INFO] [B200] Loading FPGA image: /usr/share/uhd/images/usrp_b205mini_fpga.bin...
[INFO] [B200] Operating over USB 3.
[INFO] [B200] Initialize CODEC control...
[INFO] [B200] Initialize Radio control...
[INFO] [B200] Performing register loopback test...
[INFO] [B200] Register loopback test passed
[INFO] [B200] Setting master clock rate selection to 'automatic'.
[INFO] [B200] Asking for clock rate 16.000000 MHz...
[INFO] [B200] Actually got clock rate 16.000000 MHz.
  _____________________________________________________
 /
|       Device: B-Series Device
|     _____________________________________________________
```

```
|    /
|   |       Mboard: B205mini
|   |   serial: 31BBF5F
|   |   name: B205i
|   |   product: 30522
|   |   revision: 3
|   |   FW Version: 8.0
|   |   FPGA Version: 7.0
```

Range Networks RAD1

You will need to go into the TransceiverRAD1 directory and run the following script right before running the OpenBTS-UMTS binary in the apps directory (OpenBTS-UMTS/apps/):

```
sudo ./clkit_61_44mhz.sh
```

Setup

The OpenBTS-UMTS application will be placed into the /OpenBTS-UMTS directory. Before running it, there are several other support steps necessary. The first is to instantiate some run-time artifacts needed by OpenBTS-UMTS, including the initial database:

```
sudo mkdir /var/log/OpenBTS-UMTS
```

Create the configuration database with this command:

```
sudo sqlite3 /etc/OpenBTS-UMTS/OpenBTS-UMTS.db ".read OpenBTS-UMTS.example.sql"
```

You will need to setup forwarding in iptables to properly forward data between your devices, your host machine:

```
sudo iptables -t nat -A POSTROUTING -o <INTERNET INTERFACE> -j MASQUERADE
```

And also tell the kernel that IP forwarding is allowed:

```
echo 1 | sudo tee /proc/sys/net/ipv4/ip_forward 1>/dev/null
```

(Optional) Using comp128

Checkout and install the `libcoredumper` library:

```
$ git clone https://github.com/PentHertz/libcoredumper
$ cd libcoredumper
$ ./build.sh
$ cd coredumper-1.2.1#
$ ./configure
$ make && make install
```

Then install checkout `subscriberRegistry` module and make the project:

```
$ git clone https://github.com/PentHertz/subscriberRegistry
$ git submodule init
$ git submodule update
$ ./autogen.sh
$ make
$ sudo make install
```

Copy sipauthserve's comp128 to run-time directory of OpenBTS-UMTS

```
sudo cp /OpenBTS-UMTS/comp128 /OpenBTS-UMTS/
```

Using OpenBTS-UMTS with the LimeSDR (experimental)

There is an experimental patch for LimeSDR that uses SoapyUHD to run OpenBTS-UMTS on the LimeSDR. You can try it by installing the following dependencies:

```
$ sudo apt install uhd-soapysdr libsoapysdr-dev soapysdr-module-lms7 limesuite
liblimesuite20.10-1 liblimesuite-dev
```

And then make sure Soapy see's only the LimeSDR. If the audio devices is displayed as follows you will have to remove it:

```
# SoapySDRUtil --find
#######################################################
##     Soapy SDR -- the SDR abstraction library     ##
#######################################################
```

```
[ERROR] avahi_client_new() failed: Daemon not running
[ERROR] avahi_client_new() failed: Daemon not running
[INFO] [UHD] linux; GNU C++ version 11.2.0; Boost_107400; UHD_4.1.0.5-3

RtApiPulse::DeviceInfo pa_context_connect() failed: Connection refused

ALSA lib pcm_dmix.c:1032:(snd_pcm_dmix_open) unable to open slave

RtApiAlsa::getDeviceInfo: snd_pcm_open error for device (default), No such file or
directory.

ALSA lib pcm_dsnoop.c:601:(snd_pcm_dsnoop_open) unable to open slave

RtApiAlsa::getDeviceInfo: snd_pcm_open error for device (default), No such file or
directory.

Found device 0
  default_input = False
  default_output = False
  device_id = 8
  driver = audio
  label = hw:USB Audio,0

Found device 1
  default_input = False
  default_output = False
  device_id = 9
  driver = audio
  label = hw:USB Audio,1

Found device 2
  default_input = False
  default_output = False
  device_id = 10
  driver = audio
  label = hw:USB Audio,2

Found device 3
  addr = 1d50:6108
  driver = lime
  label = LimeSDR-USB [USB 3.0] 9072C00D5261F
  media = USB 3.0
  module = FX3
  name = LimeSDR-USB
  serial = 0009072C00D5261F
```

Remove it as follows:

```
sudo apt remove soapysdr0.8-module-audio
```

Running OpenBTS-UMTS

After successfully building and configuring, you are ready to launch OpenBTS-UMTS:

```
cd /OpenBTS-UMTS
sudo ./OpenBTS-UMTS
```

Several useful commands are available for debugging the packet-switched OpenBTS-UMTS application. Launch the OpenBTS-UMTS CLI to manipulate and configure your UMTS installation.

```
sudo /OpenBTS-UMTS/OpenBTS-UMTSCLI
```

FAQ

Regarding troubleshoot SDR1, YMMV

My data stream is always dropping, what can I do?

Try increasing the transceiver buffer size. The default is 1e6, but this can be increased depending on your available hardware.

```
config UMTS.RLC.TransceiverBufferSize 5000000
rawconfig UMTS.RLC.TransceiverBufferSize 100000000
```

Clone this wiki locally

```
https://github.com/PentHertz/OpenBTS-UMTS.wiki.git
```