The OpenBTS Project

David A. Burgess, Harvind S. Samra
Kestrel Signal Processing, Inc.
Fairfield, California

October 31, 2008

Chapter 1

About The Project

1.1 What is the OpenBTS Project?

The OpenBTS Project is an effort to construct an open-source Unix application that uses the Universal Software Radio Peripheral (USRP) to present a GSM air interface ("Um") to standard GSM handset and uses the Asterisk VoIP PBX to connect calls. This is in fact very different from a conventional GSM BTS, which is a dumb device that is managed externally by a basestation controller (BSC) and connects calls in a remote mobile switching center (MSC). Because of this important architectural difference, the end product of this project is better referred to as an *access point*, even though the project is called "OpenBTS".

1.2 Why Build an Open Source GSM Stack?

The combination of the ubiquitous GSM air interface with VoIP backhaul could form the basis of a new type of cellular network that could be deployed and operated at substantially lower cost than existing technologies. Since these new hybrid networks are not readily compatible with legacy networks, and since radical two-tier pricing would be disruptive for existing carriers, we are not likely to see this kind of innovation from the conventional telecom community. This is the primary motivation for starting this project: a vision of truly universal telephone service.

The inspiration for this project came from a simultaneous recognition of these elements from prior experience[1] with GSM, software radios, VoIP and sustainable power systems:

- The USRP can be readily adapted as a GSM transceiver and the hardware can be reworked to give a carrier-grade radio for use in a software BTS.

[1]The founders of this project actually come from a signals intelligence background. They've been working with wideband digital radios and looking at telecoms networks as "black boxes" for a total of 25 years, giving them a different mental model of GSM than you might find within the cellular industry. For the most part, they've only cared about the air interface, and with VoIP that's all we really need anymore.

- We know from prior experience that a GSM protocol stack for basic telephony can probably be implemented in less than 15,000 lines of C++, including the software radio. A useful basestation can probably run in real time on a 1.5 GHz 32-bit CPU. L1 is textbook radio and FEC algorithms. L2 is simplified HDLC. L3 is fully defined in GSM 04.08 and ITU-T Q.931. It's actually not that complicated, once you understand the specifications.
- Most telephone switching functions, calling features and even mobility management functions can be moved into a software PBX like Asterisk, eliminating the need for most of the network infrastructure (HLRs, MSCs, etc.).
- Plenty of functional GSM handsets are discarded in rich countries every year that can be reused in developing countries.
- With economical software design and good hardware engineering, it should be possible to run a low-capacity GSM cell from solar panels or micro wind turbines, drastically reducing the cost of service in off-grid environments.[2]

Early in this project some of us had a conversation with an executive from a well-known (but NDA-ed and anonymous) cellular carrier. He asked, "Lowering costs sounds good, but how does this leverage my existing network?" Our answer was, "It doesn't. That's not the point. This is leapfrogging." He then asked, "Has Ericsson put a horse's head in your bed yet?"

1.2.1 GSM is Old and Boring. Why Not CDMA?

GSM is a good choice precisely because it is old and boring. Everyone knows it works and 80% of the world's carriers are still using it. It's a proven technology that is well-suited to the target application. The specification is publicly available and in a few more years most of the essential patents will expire.

CDMA physical layers are too complex for an inexpensive all-software radio and do not scale well for low-capacity cells. CMDA capacity comes in increments of 50 or more subscriber lines and the lowest layers of your radio must process all of that bandwidth whether you intend to use it or not. By contrast, GSM capacity comes in increments of 7-8 lines and a well-managed radio can even ignore inactive parts of the signal. Beyond the technical issues, IS-95-style CMDA (including cdma2000) is tightly controlled intellectual property. You can't even get a copy of the specification without signing an NDA and paying several hundred dollars.

1.2.2 What About GPRS/EDGE and UMTS?

Future versions of the OpenBTS may well support GPRS and EDGE. GPRS, when available, should be a software-only upgrade for any installed OpenBTS system. EDGE support may require additional computational resources but the additional software is not complex, at least when compared to the rest of the BTS. UMTS is a radically different CDMA-style physical layer and well outside the current scope of this project.

[2] One of the founders of the project runs most of his house on solar power.

All that said, let's walk before we run by implementing basic voice services on a single-ARFCN GMSK BTS.

1.2.3 What's Wrong with WiFi, WiMax, WiWhatever?

There are a lot of people out there who would rather *not* blanket Africa with WiWhatever than find a way to make GSM dirt cheap.[3] It's sexier to talk about the newest air interface and that talk gets a lot a buzz, but the truth is that that WiWhatever is poorly suited to mobile telephony.

WiFi range is far too short for mobile coverage in rural areas. For example, you're not going to cover 700 square miles with a single WiFi tower, but that's exactly what GSM was made to do. If access points are connected through different ISPs, handovers will be unreliable. The phones are expensive and power-hungry, and compared to GSM they always will be. WiFi may become a decent technology for semi-mobile telephones in dense urban areas, but it's not a mobile phone standard and it's not well-matched to the rural cellular application.

WiMax has most of the problems of WiFi. To make matters worse, most WiMax bands don't penetrate structures very well. Most WiMax deployers are planning to solve this problem by saturating large buildings with small access points. That's fine in Manhattan and London, but we don't see anyone putting femtocells in a million houses when those households couldn't afford phones in the first place.

More important than all of that is to remember the goal of the OpenBTS: universal telephone service. Our project philosophy is that it is much better to give people basic telephone service with an upgrade path to 250 kb/sec EDGE than to generate a lot of hype over a scorching fast broadband technology that can probably never be truly universal. Don't let the perfect be the enemy of the good.

WiWhatevers do have a place in OpenBTS, though: backhaul. The standard OpenBTS backhaul is likely to be redundant-path point-to-point WiFi or WiMax.

1.3 Legalities

Every project has the potential for abuse. Let's avoid that.

1.3.1 Intellectual Property

We can bet this project will get a lot of intellectual property scrutiny from the vindictive employers and ex-employers of every experienced telecom engineer who contributes. To limit their use of the legal system as an instrument of harassment and abuse, and to prevent genuine bad acts, please observe these guidelines:

[3] We say "not" because none of those people who talk about it are really going to do it because it's not a realistic goal.

- *Do not* insert proprietary code into this project. Any code identified as proprietary will be removed and the person who inserts it may be barred from the project.

- *Do not* insert ETSI material that is subject to non-disclosure agreements, specifically authentication and encryption algorithms. While private developers are free to acquire these algorithms and include them in controlled distributions, this material would jeopardize the open source status of the project.

- *Do* comment generously and reference specific paragraphs of GSM and ITU-T specifications to show the obvious relationship between the source code and the public specifications.[4]

While the project maintainers will make every effort to ensure that the OpenBTS is strictly non-proprietary and that all licensing requirements are met, this is not an indemnification. We do our best but guarantee nothing. Research this code well before using it in your own commercial applications.

Licensing

The OpenBTS software is being produced under the GPLv3 license.

1.3.2 The FCC (or Ofcom, or whatever they call it where you live)

Radio transmissions are regulated everywhere in the world. In the United States, these regulations are laid out in CFR 47 and enforced by the Federal Communications Commission (FCC), but every country in the world has functional equivalents. Before turning on any radio equipment or using any of this software you are *strongly* urged to know your local telecom and radio regulations. Broadly speaking:

- If you are transmitting in a licensed band and your transmissions are detectable more than a few meters away then you are probably breaking the law.

- If you are receiving signals from a licensed public network in sufficient volume to comprehend conversations, determine the contexts of data transmissions or determine the identities of users then your are probably breaking the law.

- If you are outside of North America and Europe and using VoIP over a public network, you may be breaking the law.

[4]Notice how careful we are to point out obvious relationships between the OpenBTS design and specific sections in GSM documents? That's because the founders of the project already have an ex-client trying, through both legal and extra-legal means, to shut down their previous attempt to build a GSM BTS.

1.3.3 Criminal Activity

Any discussions of the exploitation of public telephone networks for criminal purposes will be removed and forwarded to appropriate authorities. Go build your IMSI-catchers and interceptors somewhere else.

Chapter 2

Understanding GSM

The GSM standard is large and complex, but it is also modular and can be understood one module at a time if you know where to find the documentation. The purpose of this chapter is not to explain GSM but to give a very brief "big picture" sketch of the system and references to documents that give more complete and authoritative information. This chapter focuses almost entirely on the air interface between the mobile station (MS) and the basestation (BTS). This interface is called "Um" in GSM terminology because it is intended as a functional equivalent of the ISDN "U" interface.

2.1 Standards and Resources

Standards documents can be downloaded for free as needed from the ETSI 3GPP site:

`http://webapp.etsi.org/key/queryform.asp`

The GSM specification will look like a cookbook once you understand it, but it is not a tutorial and is a hard place to start. For that, a good choice may be An Introduction to GSM, Redl, Weber & Oliphant, ISBN 0-89006-785-6. There are plenty of good university lectures available on the internet, too.

2.2 The Radio Layer, FEC and TDM (GSM 05.xx)

The GSM 05.xx series of specifications describes the GSM radio layer, with GSM 05.01 as an introduction.

GSM 05.05 defines bands, frequencies and carrier spacing. The GSM radio channel has a bandwidth of 270.833 kHz and a spacing of 250 kHz. Since adjacent channels overlap the standard does not allow adjacent channels to be used in the same cell. Uplink and downlink bands are generally separated by 50 MHz. Uplink/downlink channel pairs are identified by an index called the Absolute Radio Frequency Channel Number (ARFCN).

GSM 05.04 defines the modulation. GSM uses GMSK with a 270,833.333 Hz symbol rate. The channel is time-domain multiplexed into 8 timeslots, each with a duration of 156.25 symbol periods. These 8 timeslots form a frame of 1,250 symbol periods. The capacity associated with a single timeslot on a single ARFCN is called a physical channel (PCH).

Each timeslot is occupied by a radio burst with a guard interval, two payload fields, tail bits, and a midamble (or training sequence) for channel estimation and burst detection. The lengths of these fields vary with the burst type. The Normal Burst (NB) has a 8.25-symbol guard period, two 57-bit payload fields, two 3-bit tail bit fields and a 26-bit midamble. There several other burst formats, though. Bursts that require higher processing gain for signal acquisition have longer midambles. The random access burst (RACH) has an extended guard period to allow it to be transmitted with incomplete timing acquisition. Burst formats are described in GSM 05.02 5.2.

GSM timing is driven by the serving BTS. All clocks in the handset, including the symbol clock and local oscillator, are slaved to signals received from the BTS. (See GSM 05.10.) Unlike IS-95 CDMA, the BTSs in the GSM network can be asynchronous. All timing requirements in the GSM standard assume that clocks are derived from a stratum-3 OCXO, because that was the standard clock technology at the time the specifications were written.

2.3 Logical Channels and Their Layers (GSM 04.xx)

Each PCH is further time-multiplexed into logical channels (LCHs). This multiplexing is driven by the BTS frame clock and generally follows a 51- or 26-frame structure called a multiframe. The numbers and types of LCHs sharing a PCH multiframe and their multiplexing rules are described in GSM 05.02, specifically in Section 6.4 and the tables of Clause 7. The channel types themselves are outlined in GSM 04.03 Section 4.

2.3.1 LCH Types

Each LCH can be thought of as a transport stream between the BTS and the mobile station (MS) with specific LCH types used for specific purposes.

- Synchronization Channel (SCH). A downlink-only broadcast channel that carries an abbreviated system identification and the current value of the BTS frame clock. This channel is used to sync the MS to the BTS frame clock and multiplexing structures. This is the first channel acquired by the handset.

- Frequency Correction Channel (FCCH). A downlink-only broadcast channel that produces a fixed tone in the modulator. This channel is used by the MS to estimate the carrier frequency and lock the handset's local VCXO to the BTS's OCXO.

- Broadcast Control Channel (BCCH). A downlink-only broadcast channel that carries detailed information about the BTS identity and configuration.

- Common Control Channel (CCCH). A downlink-only unicast channel that is used to page MSs and set up point-to-point channels for actual signaling transactions.
- Random Access Channel (RACH). An uplink-only shared channel used by MSs to request service from the BTS. Successful random access attempts are answered with resource allocation messages on the CCCH.
- Standalone Dedicated Control Channel (SDCCH). A bidirectional point-to-point channel used for call set-up, registration, SMS delivery and other control signaling when a call is not yet connected. The SDCCH is a functional equivalent of the IDSN D-channel, sometimes called Dm (along with the FACCH).
- Traffic Channel (TCH). A bidirectional conduit for user voice and data (ie, user traffic). The GSM traffic channel is the functional equivalent of the ISDN B-channel, sometimes called Bm.
- Fast Associated Control Channel (FACCH). A bidirectional point-to-point control channel used for signaling during a connected call. The FACCH is a blank and burst channel that operates by stealing bandwidth from the TCH. The FACCH is a functional equivalent of the IDSM D-channel and sometimes called Dm (along with the SDCCH).
- Slow Associated Control Channel (SACCH). A bidirectional point-to-point channel used for auxiliary signaling during any transaction where a TCH/FACCH or SDCCH is present. The SACCH is normally used to carry system information and measurement reports related to handover planning and link status. It can also carry SMS during an in-progress call. Like the SDCCH and FACCH, the SACCH is modeled after the ISDN D-channel.

2.3.2 The Layers of an LCH

Although GSM predates OSI, each GSM LCH is organized into layers that roughly follow the OSI model.

Radio Modem and TDM (in L1)

These were described in Section 2.2.

Forward Error Correction (in L1)

GSM 05.03 describes the forward error correction on the various GSM channels. Typically, this error correction coding is in three stages.

1. Interleaving. In most GSM channels, a higher-layer data frame is spread over 4 radio bursts with an interleaving pattern according to the rules of GSM 05.03 Sections 4.1.4 and 4.1.5.

Some channels (TCH/FACCH) use 8-burst diagonal interleaving as per GSM 05.03 Sections 3.1.3 and 3.1.4. In some channels (RACH, RACCH and SCH) each protocol element maps to a single radio burst and there is no interleaving.

2. Convolutional Encoding. Most GSM channels use a rate-1/2 convolutional encoder described in GSM 05.03 Section 4.1.3.

3. Block Coding. All GSM channels use some type of block code. Some channels use simple CRC-type parity check codes. Most channels use a Fire code that can also be used for burst-error correction, described in GSM 05.03 Section 4.1.2.

Data Link (L2)

The function of L2 is to segment higher layer messages into L2 frames and to provide for ordering and retransmission of those frames. The GSM data link layer is called LAPDm and documented in GSM 04.05 and 04.06. LAPDm borrows heavily from ISDN LAPD (ITU-T Q.921) and GSM 04.06 is written assuming the audience has access to both documents. Both LAPD and LAPDm are derived from HDLC (ISO-13239), a protocol used in frame relay, X.25 and old IBM mainframe networks. As the name would suggest ("Link Access Protocol on the D-Channel"), LAPDm is most active on the FACCH and SDCCH, since these are functional equivalents of the ISDN D-channel. It is also active on the SACCH when text messages are delivered for an in-progress call. On most other channels, LAPDm has little effect but to format a header on the L2 frame.

Signaling Layer (L3)

GSM L3 defines the protocol messages and control transactions between the BTS and MS. There are three sub-protocols defined in L3:

- Radio Resource (RR) – for management of radio channels. This is a GSM-specific protocol defined in GSM 04.08.

- Mobility Management (MM) – for subscriber authentication and routing calls to specific geographic areas. This is GSM-specific protocol defined in GSM 04.08.

- Call Control (CC) – connection management for speech telephony. GSM call control is defined in GSM 04.08 but closely follows that of ISDN (ITU-T Q.931). Both documents are needed for a complete understanding.

2.4 Typical Access Sequence

Here is the typical sequence of events when an MS powers up in a new cell and registers in that cell, as described in GSM 04.08 Sections 3 and 4:

1. The MS powers up and searches its supported bands for the ARFCN with the strongest signal. This signal is assumed to be the beacon of a nearby cell.
2. The MS searches (with a correlator) for the SCH on the selected ARFCN. (If no SCH is found, the MS continues to try other ARFCNs according to their power.)
3. The MS decodes the SCH to get BTS timing and syncs its local frame clock with that of the BTS. (The content of the SCH message is described in GSM 04.08 Section 9.1.30.)
4. Having the correct frame timing, the MS can now demultiplex the BCCH. The BCCH identifies the carrier and provides detailed information on the services offered by the BTS and the multiplexing configuration of the CCCH. This information is carried in a series of System Information Messages, described in several subsections of GSM 04.08 Section 9.1.
5. The MS starts decoding the CCCH and sends access requests on the RACH. Each RACH message occupies a single radio burst and contains a random tag. The MS sends up to 8 RACH bursts separated by random delays of 1-2 seconds while checking the CCCH for a response. The content of the RACH burst is documented in GSM 04.08 Section 9.1.8.
6. The BTS receives a RACH burst from the MS and responds with a channel assignment message (Immediate Assignment, GSM 04.08 Section 9.1.18) on the CCCH. In this message the BTS echoes back the tag and timing from the corresponding RACH burst so that the MS can recognize it. The message assigns the SDCCH for a transaction.
7. The MS receives the immediate assignment on the CCCH and changes its multiplexing and possibly its ARFCN to match the parameters of the message.
8. The MS switches over to the assigned SDCCH, waits (with a brief timeout) to detect an SDCCH in the idle state. Upon verifying the SDCCH, the MS sends a Location Updating Request (GSM 04.08 Section 9.2.15) containing some form of mobile identity.
9. Assuming the BTS accepts the MS, it responds with a Location Updating Accept (GSM 04.08 Section 9.2.13). The MS and BTS then close the SDCCH and release the associated radio resources.
10. The MS is now "camped" on the BTS and continues to monitor the CCCH for Paging Request messages (GSM 04.08 Sections 9.1.22-9.1.24) associated with incoming (mobile-terminated) calls or text message deliveries.

2.5 Control and Switching

2.5.1 In a Conventional GSM Network

In its original form, the GSM BTS is a fairly dumb radio with no resource management or switching functions. The BTS connects to a basestation controller (BSC) through an interface called Abis, which is essentially an encapsulation of the L2-L3 interface. The BSC manages the radio functions of the BTS and routes all call connection signaling and user traffic to a mobile switching center (MSC). It is the MSC that actually connects calls and tracks users in the network.

2.5.2 In a Hybrid VoIP-GSM Network

In the OpenBTS project we are going to deviate from the standard GSM architecture by moving all radio control functions and most call switching functions either into the access point itself or into a VoIP PBX running locally to the access point. Local termination of L3 is unusual for GSM but not unique, since all IMSI-catchers, most "network-in-a-box" devices and some femtocell designs also terminate L3 locally. In the hybrid GSM/VoIP vision, the resulting cellular system is no longer a hierarchy of different functions but a peer-to-peer network.

Chapter 3

Software Architecture of the OpenBTS

This chapter describes the overall architecture and implementation plan of the OpenBTS.

The general shape of the OpenBTS is a collection of parallel "logical channels", defined in GSM 04.03. Each channel, in turn, is built from layers that roughly follow the OSI standard model. (We say "roughly" because GSM predates the OSI model.) GSM 04.01 Section 7 gives an initial overview of these layers.[1]

As per GSM 04.01 Section 7.3, the layers of the air interface Um are:

- L1, "PHYSICAL LAYER", the radio modem, time-division multipexing, and error-correcting coding, GSM 04.04 and GSM 05.xx series
- L2, "DATA LINK LAYER", link layer addressing, segmentation and retransmission (LAPDm), GSM 04.05 and 04.06, ITU-T Q.921.
- L3, "L3", signaling and connection management, GSM 04.07, 04.08, 04.10, 04.11, 04.12 and ITU-T Q.931.

When describing these layers we will refer to the "low" side of the layer as the side the interfaces closer to the hardware and the "high" side as the side that interfaces to higher-level functions. In a BTS, the the low-to-high direction is "uplink" and the high-to-low direction is "downlink". In the MS, those labels are reversed.

The typical "upstream" data flow in the GSM air interface of the OpenBTS, is:

1. Radio bursts arrive at the USRP and are digitized. The resulting samples are transferred to the transceiver software in the host CPU in time-tagged USB packets, using the standard USRP interface.

[1]Some people are ignorant enough to think this is a trade secret. That's pretty funny until they sue you for misappropriating their supposed intellectual property.

2. The transceiver syncs the USRP timetags with the GSM master clock, isolates each radio burst and demodulates it into a vector of symbol likelihoods ("soft symbols"). The modulation format is defined in GSM 05.04 and the burst formats are described in GSM 05.02 Section 5.2.

3. The soft symbol vector for each radio burst is timetagged with the GSM frame clock and transferred to the GSM stack via a datagram interface.

4. In the GSM stack, the TDM sublayer (of L1) demultiplexes each bust according to its timetag and sends it to the appropriate logical channel.

5. The logical channel passes each burst into its L1 FEC processor according to the rules of GSM 05.02.

6. The L1 FEC processor performs the FEC decoding described in GSM 05.03. The output is a sequence of L2 frames taken by the logical channel and sent up to an L2 processor.

7. The L2 processor runs the LAPDm state machine that performs acknowledgments, retransmissions and segmentation. This state machine is defined implicitly in GMS 04.06 and given explicitly in ITU-T Q.921. When an incoming L3 frame has been verified and assembled, it is placed into a queue for consumption by L3. In the course of operation, LAPDm also injects L2 frames into the downstream flow for acknowledgment and retransmission requests.

8. In L3, a dispatch function determines the message protocol and type and calls the appropriate control function to deserialize the message and act on its content, generally producing an L3 response on the downlink. These control functions also interact with the outside world via SIP and other protocols.

The typical "downstream" data flow in the GSM air interface of the OpenBTS is:

1. In L3, a control function generates an L3 message, serializes the message into an L3 frame and sends it into the logical channel, which in turn passes it down to L2.

2. The L2 processor breaks the L2 frame into segments, wraps each segment in an L2 frame. Each L2 frame is sent down to L1 according to the LAPDm state machine of GSM 04.06 and ITU-T Q.921. LAPDm may also generate additional L2 frames on its own according to its acknowledgment and retransmission rules.

3. The L1 FEC processor encodes each L2 frame according to the rules of GSM 05.03, generating four outgoing radio bursts. Each radio burst is timetagged with its intended transmission time according to the TDM rules of GSM 05.02. These bursts are passed on to the TDM interface.

4. The downstream TDM sublayer is just a mutex-controlled socket interface where the radio bursts from L1 are reformatted into messages on the transceiver's datagram interface.

5. Upon arriving in the transceiver, the outgoing radio bursts are sorted into a priority queue according to transmission time. Bursts are pulled from the queue as they become ready for transmission and the modulated according to GSM 05.04. The modulated waveform samples are sent to the USRP over the standard timetagged USB interface. If no burst is ready for transmission at a given time the transceiver generates an appropriate filling sequence.

6. In the USRP the samples are converted to an analog waveform for transmission over the radio channel.

3.1 Coding: Use of Objects, Threads and C++

The OpenBTS software is written in C++ and should be portable to any 32-bit Unix system. Initial supported platforms will be Fedora and Darwin, but there is no reason for the resulting system not to be portable to other Unix variants. In the interest of simplicity, compactness and efficiency, limit our use of allocated memory, multiple inheritance, templates and standard library container to the greatest degree practical.

3.1.1 Threads

The implementation of the OpenBTS makes use of the POSIX thread library, pthreads. For those unfamiliar with threads or who have an aversion to them, this may seem frightening and strange, but the effect of this approach is to greatly simplify the code in each layer of the BTS. You can have N threads each with K states or one thread with K^N states. We choose the former. The price of threads is discipline in the use of synchronized interthread communication. *Do* use mutexes. *Do not* just use volatiles.

The general rule is that single-state transformational operations are performed with direct calls while multi-state machines, especially those with internal timeout behaviors, have their own threads and accept their inputs over FIFOs. The threads and FIFOs may be hidden inside the state machine objects. This approach allows the state machines to be decoupled, greatly simplifying the code.

3.1.2 Namespaces

The OpenBTS will use the following namespaces:

- GSM – The GSM L1-L2 stack and L3 messages.
- VoIP – The SIP stack.
- Control – The GSM/SIP hybrid control layer.

3.1.3 Optimization

From prior experience with several software radio systems, we expect that the bulk of the computational load in the OpenBTS will be in the software transceiver and L1 FEC decoding. That's because L2 frames arrive at a maximum rate of about 50 Hz while radio burst are transacted at about 200 Hz and GSM channel symbols must be processed at 271 kHz. For the rural telecom application, optimization of beacon generation and RACH detection is also important since the

BTS is expected to sit largely idle most of the time. There's not a lot of point in optimizing most of the rest of the code for speed. Instead, the rest of the system should be optimized for space to minimize cache misses in the computationally intensive code.

3.1.4 Testing and Consistency Checking

The full BTS will be a complex system that will be nearly impossible to debug if the individual components are not tested in isolation prior to integration.[2] To enforce good testing practices, we recommend the following:

- Use assert() calls liberally to insure internal consistency. Do not continue once an unresolvable error condition is identified.
- Build unit test fixtures for every component in the system. Do not check in code that does not pass the unit tests. It would be especially nice if someone could automate this testing process.
- If you know there's a potential problem in a piece of code, use a "FIXME" comment. If you leave something undone, use a "TODO" comment.
- Use the bug tracking system. That's why it's there.

3.1.5 C++ Tools

Part of what will let us build an efficient BTS quickly is careful choice of underlying tools.

Wrappers

The project uses a set of simple wrapper classes that give object-oriented interfaces for some pthreads and stdlib mechanisms. These include wrappers for threads, signals, recursive mutexes, and C time-of-day facilities. These wrappers simplify the use of these facilities by including standard allocation and initialization steps that are common across the application.

Vector Classes

The Vector template is a fixed size array that offers simple pointer-based access. The key feature of the Vector subclass is the ability to create "aliases", where multiple Vector objects share the same physical block of memory. These aliases can also have different offsets into the common block, forming subvectors. The underlying mechanism is a C array. The SoftVector is used to represent symbol likelihood vectors used in soft input and soft output Viterbi coders and decoders. The BitVector is a Vector class for serialization and deserialization of packed bitfields. The SignalVector and ComplexVector are are numeric classes used in the GMSK modem.

[2] Believe us on this one. We've seen the pain.

Interthead Classes

Most interthread communication is mediated through special container classes with built-in synchronization controls.

The InterthreadQueue class is a thread-safe pointer FIFO that supports non-blocking writes, non-blocking reads and blocking reads with a timeout. The underlying mechanism is a singly-linked list with head and tail pointers having no maximum size. The normal use of the InterthreadQueue is to allocate an object and then put the pointer into the queue. The reader is responsible for deleting the pointer. For fixed-size objects used in this manner, a special-purpose allocator can reduce overhead.

The InterthreadMap class is a thread-safe associative array that includes a blocking read that waits for the arrival of a specified key. The underlying mechanism is std::map.

3.1.6 Important C++ Object Hierarchies

The OpenBTS will use C++ object classes and inheritance. There are a few object hierarchies of particular interest:

- Layer Processor Classes. These objects implement the layers L1-L3 of the air interface.
- Data Transfer Classes. These are the objects that are used for interlayer communication. Most are BitVector subclasses.
- Logical Channel Classes. These objects define the L1-L2 logical channels between the MS and the BTS control functions.
- L3 Message and Element Classes. These are used to serialize and deserialize the messages of GSM 04.08.

Data Transfer Object Classes

The interlayer communication uses these object classes:

- RadioBurst. Carries a radio burst of GSM 05.02 Section 5.2 along with timestamp.
  - TxBurst. Used for downlink radio bursts to be transmitted. Carries hard-valued channel bits and transmit power level relative to full scale on the given ARFCN. The timestamp indicates the intended transmission time.
  - RxBurst. Used for received uplink radio bursts. Carries soft-value channel bit estimates, RSSI, and receive timing error. The timestamp indicates the arrival time of the burst.
- L2Frame. Carries an L1-L2 interlayer primitive. Optionally carries a data link layer frame as described in GSM 04.06 Section 2 and a restriction on the placement of the data in the TDM structure.

- L3Frame. Carries an L2-L3 interlayer primitive. Optionally carries the serialized bits of a single, complete L3 message from GSM 04.08 Section 9.

Logical Channel Structure and Subclasses

Following the OSI model, each logical channel is processed in three layers: L1, L2, L3. Each layer is represented by an object (a layer processor) that has read and write methods on its high and low sides. These objects are persistent over the life of the application. In the normal course of the OpenBTS application, the destructors for these classes should never be invoked.

The layer processor classes are:

- L1FECEncoder. Accepts L2Frame objects, encodes them into TxBurst objects. Algorithms defined in GSM 05.03.
- L1FECDecoder. Accepts RxBursts from the TDM sublayer and processes them into L2Frame objects. Algorithms defined in GSM 05.03.
- L1FEC. A container for a channel's L1FECEncoder and L1FECDecoder.
- L2LAPDm. The L2LAPDm transacts L2Frame objects with L1 and L3Frame objects with L3. Implements the LAPDm state machine as described in GSM 04.06 and ITU-T Q.921.
- L3StateMachine. The L3StateMachine transacts L3Frame objects with L2. Implements the protocols of GSM 04.08 and also interfaces with SIP entities.

- BCH (broadcast channels)
  - SCH
  - FCCH
  - BCCH
- NDCCH (non-dedicated control channels)
  - CCCH (downlink unicast)
  - RACH (uplink shared)
- DCCH (dedicated control channels, Dm)
  - SDCCH
  - SACCH
  - FACCHF (full rate)
- TCHF (full rate speech, GSM 06.10, Bm)

Many traffic channel classes are ignored here, as are half-rate channels. They are not a priority in a first release.

L3 Message Classes

The L3Message class carries one of the messages of GSM 04.08 Section 9. Each L3 subprotocol has a subclass. Each member object has serialize, deserialize and accessor methods.

3.2 Control Functions

Control functions carry out the various procedures described in GSM 04.08 Sections 3-5. These functions are called in L3. Planned transactions in the first implementation include:

- access grant (GSM 04.08 Section 3.3.1)
- paging (GSM 04.08 Section 3.3.2)
- location updating (GSM 04.08 Section 4.4)
- mobile-terminated call setup (GSM 04.08 Section 5.2.1)
- mobile-originated call setup (GSM 04.08 Section 5.2.2)
- mobile-terminated call disconnect (GSM 04.08 Section 5.4.4)
- mobile-originated call disconnect (GSM 04.08 Section 5.4.3)

This is the minimum transaction set to support telephony. These transactions are run by these top-level control functions:

- AccessGrantor sends out SDCCH Immediate Assignments on the CCCH. It is invoked from the RACH dispatcher and returns when the assignment has been delivered. Its arguments are the tag and timestamp of a RACH Channel Request message.
- Pager sends out paging messages. It is invoked from the SIP stack whenever an INVITE message arrives and returns when the subscriber IMSI has been enqueued for paging on the CCCH. Its argument is a reference to an IMSI.
- LocationUpdater runs the registration transaction of GSM 04.08 Section 4.4. It is invoked by the SDCCH dispatcher when a Location Updating Request is received and returns when the transaction is complete. Its argument is a reference to an SDCCH LogicalChannel where the transaction is carried out.
- MTCConnector starts a mobile terminated call and returns when the call is cleared. It is invoked by the SDCCH dispatcher when a Paging Response is received. It continues the SIP transaction started by the INVITE message that triggered the corresponding Paging Request. Its argument is a reference to an SDCCH LogicalChannel where the transaction will start. It allocates a TCH/FACCH as part of the transaction. It calls CallManager once the call is connected.

- MOCConnector starts a mobile originated call (MOC) and returns when the call is cleared. It is invoked by the SDCCH dispatcher when a CM Service Request is received. Its argument is a reference to an SDCCH LogicalChannel where the transaction will start. It allocates a TCH/FACCH as part of the transaction. It calls CallManager once the call is connected.
- CallManager runs a call once the setup transaction is complete. It processes any L3 messages that arrive while the call is in progress and invokes functions as needed for the call disconnection state machines.

Most control layer functions return boolean values indicating whether or not the transaction ended during the function call. For example, a function for some call control sub-procedure will return a boolean indicating whether or not the call was cleared during the sub-procedure. Most control layer functions take a LogicalChannel reference as an argument

3.3 Time Domain Mutliplexing (TDM) Sublayer

The TDM sublayer forms the interface between the physical channels on the transceiver interface and the logical channels in the higher layers. It implements the rules of GSM 05.02. All LCHs on an ARFCN share a common TDM sublayer. In the uplink side, the TDM sublayer accepts all of the RxBurst objects from a single ARFCN and demultiplexes them according to their frame count timestamps into a table of LogicalChannel objects. In the downlink side, the multiplexing layer is a pass-through to the transceiver interface, but with mutex controls and serialization.

3.4 Software Transceiver

The software transceiver includes the subband tuner, GMSK modems and the master GSM symbol clock for the air interface. Its low side interface transfers raw baseband samples to and from the USRP. Its high side interface transfers RxBurst and TxBurst objects to and from the GSM stack and transacts control commands and clock values. To compensate for latency in the stack and interface, the GSM frame clock value reported to the GSM stack is advanced slightly. The timing reference for the GSM symbol clock is the sample clock in the USRP, as is standard in most software radio designs.

For the uplink, the software transceiver has information about which timeslots and frame positions carry active channels and it demodulates only those active parts of the signal. It timestamps each TxBurst according to the frame clock value when the burst arrived.[3]

For the downlink, the timestamp on an TxBurst indicates the time at which a radio burst is to be transmitted. If a downlink radio burst is not available for transmission when the GSM clock reaches a given value, the transceiver generates an appropriate filler pattern based on the clock value. TxBursts can sent over this interface out of order, with the transceiver interface sorting them prior to transmission.[4] If the TxBurst's indicated time has passed, the burst is discarded

[3]This is similar in principle to RTP or the timecoded sample packet stream on the USRP's own UDP interface.
[4]This is similar in principle to timecoded packetized video formats such as M2TC.

and the clock advance value is adjusted to prevent further late bursts. This adjustment is in one direction and quickly settles to an appropriate value for the system.

Because the software transceiver links with the GPL'd USRP driver, it is compiled as a separate application and communicates with the rest of the GSM stack over a UDP interface.[5] Expressed in terms of a base port, B, the UDP ports used for the transceiver interface are:

- master clock interface on port B
- control interface for ARFCN N on port $B + 2N + 1$
- data interface for ARFCN N on port $B + 2N + 2$

The protocol on the master clock interface has a single message that periodically indicates the current value of the BTS master clock, with an additional advance to compensate for latency. This message is an unacknowledged indication from the transceiver to the GSM stack. These are human-readable ASCII messages sent once per superframe and whenever the clock advance value is adjusted.

The protocol on the per-ARFCN control interface has commands for power control, tuning and configuration of the channel combination on each slot. Each command has a corresponding response for integrity on the UDP link. The are human-readable ASCII messages.

The protocol on the per-ARFCN data interface uses binary messages to convey RxBurst and TxBurst objects with one burst per UDP packet to minimize latency. There are no acknowledgments on this interface because there is no point in retransmitting packets in a low-latency realtime link.

Eventually, we will replace to USRP driver with non-GPL code and replace this socket protocol with a direct method interface just like the interface on every other layer and sublayer object in the system.

3.5 Global GSM Object

There is a single global object of the class GSM::Configuration, called gBTS, that acts as a collection point for the complete state and configuration of the access point's GSM stack. This object is created during initialization and never destroyed.

[5] We use UDP instead of TCP because is it more efficient, easier to manage and its connectionless nature allows us to independently restart different components of the system. Our communication is message-oriented and better suited to UDPs datagrams to than to TCPs streams. Packet loss on localhost and LAN connections is very rare. We are not using Unix datagrams because most kernels are configured with datagram queues that are too short for this application. In the long term, though, we want to integrate the transceiver and GSM stack into a single application.

3.5.1 Global Configuration Information

The global object gBTS keeps all of the BTS configuration parameters and state for the cell, including ARFCN set, cell ID, LAC, etc. This information is used to fill system information messages on the BCCH and SACCH and is readily available to the control layer.

3.5.2 LCH Creation, Allocation and Deallocation

The gBTS object tracks all of the existing LogicalChannel objects in the system in an allocation pool. The allocator includes methods to create new LogicalChannel objects and add them to the pool as the BTS's channel combinations are configured on start-up. Once created, these LogicalChannel objects are persistent over the life of the application.

When a control layer function determines that it needs a new logical channel for a transaction, it invokes the channel allocator to find an available LogicalChannel object of the proper type in the pool. Channel availability is defined by a set of Z.100-type timers in L1: T3101, T3107, T3109 and T3111, described in GSM 04.08 Section 11.1.2.[6] If any timer is expired, the channel is available. Once identified, the channel is opened, reseting the channel state and starting T3101 or T3107. This timer change moves the channel to the non-available state. The channel is then given to the calling control layer function for use in a transaction.

Normally, at the end of the transaction the channel is closed from above with a "release" primitive that passes down to L1, starting T3111. Once T3111 expires, the channel becomes available for reuse. In an abnormal release (a dropped connection), one of the other timers will expire instead. T3107 or T3101 will expire if the MS fails to pick up an assigned channel at the start of a transaction. T3109 will expire if the uplink signal is lost during a transaction. Any any case, one of the timers *will* expire, moving the channel to the available state.

It should be stressed that this mechanism does not create or destroy LogicalChannel objects. It manages a pool of long-lived objects that are recycled for different transactions, just like the radio channels themselves.

3.6 Use of SIP and Asterisk

One of the keys to simplifying the OpenBTS is to push as much of the control layer as possible into the Asterisk PBX. To that end, we will use Asterisk for nearly all call control functions and as an integral part of mobility management. The simplest way to do this in a first release is to use subscriber IMSIs as SIP user names and to present each GSM handset to Asterisk as a SIP client. The OpenBTS control layer (L3) largely exists to perform mapping operations:

- GSM location updates get mapped to SIP registrations.

[6] This list does not include timers associated with handovers since we're not supporting handovers in the first release.

- Call connection transactions get mapped to corresponding SIP transactions. Since GSM call control is very similar to H.323 and ISDN/Q.931 call control, there's very little that is mysterious or novel about this mapping.
- GSM traffic channels get mapped to RTP channels. Again, the mapping is obvious and the only tricky piece of engineering is latency control.

Only radio resource operations are fully terminated in the OpenBTS itself, with the RR control layer providing the functional equivalent of a BSC. Everything else maps to SIP and eventually terminates in Asterisk, with the Asterisk network providing the functional equivalent of an MSC. This is "network in a box" with Asterisk replacing the SS7 parts.

3.7 Supported Transactions and Messages

The first goal is to build the minimum L3 and control layer to support mobile originated and mobile terminated calls with subscribers identified by IMSI. This first release will *not* support TMSIs, frequency hopping, SMS, GPRS or VGCS. Once we reach this first release, though, many of these features can be added with modest effort.

All specification references here are from GSM 04.08. In all of these messages, optional and conditional elements need not be supported in the first release unless otherwise indicated. Rest octets need not be supported in the first release.

- Beacon Generation (3.2.2). The beacon identifies the BTS to the handsets in the area. Frequency list elements in these messages need only support the bit map 0 (10.5.2.13.2) and variable bitmap (10.5.2.13.7) formats.
  - System Information Type 1 (RR, 9.1.31), encode.
  - System Information Type 2 (RR, 9.1.32), encode.
  - System Information Type 3 (RR, 9.1.35), encode.
  - System Information Type 4 (RR, 9.1.36), encode.
- RR Connection Establishment (3.3.1). This procedure is used to establish the SDCCH at the start of most transactions.
  - Channel Request (RR, 9.1.8), decode. This is the payload of the RACH burst.
  - Immediate Assignment (RR, 9.1.18), encode. The packet channel description and starting time elements need not be supported. The mobile allocation element need not be supported and can simply be coded as length 0.
  - Immediate Assignment Reject (RR, 9.1.20), encode.
  - Channel Release (RR, 9.1.7), decode and encode.
  - System Information Type 5 (RR, 9.1.37), encode. (Used on the SACCH.)
  - System Information Type 6 (RR, 9.1.40), encode. (Used on the SACCH.)

- Location Updating Procedure (4.4.4, 7.3.1)
  - Location Updating Request (MM, 9.2.15), decode. This message triggers a registration attempt with Asterisk.
  - Location Updating Accept (MM, 9.2.13), encode. This message is returned upon successful registration with Asterisk.
  - Location Update Reject (MM, 9.2.14), encode. This message is returned upon registration failure with Asterisk.
- Mobile Originated Call Establishment (7.3.2). The example in the Figure 7.10a of GSM 04.08 shows call establishment with authentication. In the open source version, we will skip the authentication steps and simply respond to the CM Service Request with a CM Service Accept or Reject.
  - CM Service Request (MM, 9.2.9), decode. This message indicates to the BTS that the phone will require ISDN-style (connection-oriented) services.
  - CM Service Accept (MM, 9.2.5), encode. This message allows us to bypass authentication and proceed directly to ISDN-style/Q.931call control.
  - CM Service Reject (MM, 9.2.6), encode. This message is sent to reject service if the call setup cannot be performed for some reason, such as an lack of available traffic channels.
  - Setup (CC, 9.3.23), decode. Simply skip unsupported elements based on T and L fields when parsing. This message triggers an INVITE message to the Asterisk server that starts the hybrid Q.931/SIP call setup procedure.
  - Call Proceeding (CC, 9.3.3), encode.
  - Assignment Command (RR, 9.1.2), encode. This message moves the transaction from the SDCCH to the TCH/FACCH.
  - Assignment Complete (RR, 9.1.3), decode. This is the first message sent from the phone on the newly assigned TCH/FACCH.
  - Alerting (CC, 9.3.1), encode. This message corresponds to the SIP 180 Ringing message.
  - Progress (CC, 9.3.17), encode. This message is needed to keep the connection alive when Asterisk delays are large. It is similar in purpose to the SIP 100 Trying message.
  - Connect (CC, 9.3.5), encode. This message corresponds to the SIP 200 OK message.
  - Connect Acknowledge (CC, 9.3.6), decode. This message corresponds to the SIP ACK message.
- Mobile Terminated Call Establishment (7.3.3). The example in the Figure 7.11a of GSM 04.08 shows call establishment with authentication. In the open source version, we will skip the authentication steps by sending a CM Service Accept or Reject after receiving the Paging Response.[7]

[7]The implication here is that the Paging Response *implies* a connection mode service request as well. This is why the message includes a classmark element.

  - Paging Request Type 1 (RR, 9.1.22), encode. This message is sent whenever a SIP INVITE message arrives from Asterisk. Because we are not yet supporting TMSIs, we have no use for the other Paging Request formats.
  - Paging Response (RR, 9.1.25), decode.
  - Setup (CC, 9.3.23), encode. This message carries the content of the SIP INVITE message to the phone.
  - Call Confirmed (CC, 9.3.2), decode.
  - Assignment Command (RR, 9.1.2), encode.
  - Assignment Complete (RR, 9.1.3), decode.
  - Alerting (CC, 9.3.1), decode.
  - Progress (CC, 9.3.17), encode.
  - Connect (CC, 9.3.5), decode.
  - Connect Acknowledge (CC, 9.3.6), encode.

- Vocoders. We will initially support the "full rate" vocoder of GSM 06.10, relying on Asterisk to run the transcoders for us.

3.8 Hardware

3.8.1 USRP

The USRP is a wideband digital radio sold by Ettus Research of Mountain View, California. It's a freak'n miracle box and without it a project like this would be impossible. The USRP is inexpensive and offers enough bandwidth to eventually support frequency hopping in the digital baseband. In the initial design of the OpenBTS, the USRP is used to provide a raw digital channel, with all modulation and demodulation performed in software on the host machine. The FPGA of the USRP is used only for subband tuning and decimation.

For cellular band operation, we use Ettus RFX900 daughter cards, but have to disable the built-in ISM passband filters.

If anyone has a good cheap source for cellular and PCS/DCS antennas with SMA connectors, please let us know.

The USRP Clock

In most GSM implementations, the symbol clock is derived from a 13 MHz "stratum 3" OCXO with a frequency accuracy on the order of 20 ppb. (A VCTCXO in the handset is slaved to the OXCO in the BTS via the FCCH.) The standard USRP clock is a low-cost 64 MHz crystal oscillator. This poses two problems for the OpenBTS:

- The USRP standard clock is not well-matched to the GSM symbol rate. In the current OpenBTS implementation, the software radio includes a resampler for rate-matching, but this is an additional computational cost that should be eliminated.
- The USRP standard clock is not stable enough for use in multi-BTS networks. The drift of the crystal oscillator may even be outside the frequency search range of some GSM phones.

Future versions of OpenBTS will assume a USRP with a modified clock, using an OCXO at a multiple of 13 MHz.

3.8.2 Analog Radio Path

This section describes everything between the USRP and the antenna.

Duplexer

For low-power testing, it's OK to use separate transmit and receive antennas, but to prevent the downlink signal from overwhelming the uplink receiver, a fielded BTS needs a good, cheap duplexer design. It would be great to have a standard duplexer design than can be built entirely from odd bits from the local hardware store.[8] Until then, we will all have take what we can get on eBay.

Transmit Power Amplifiers

Efficiency is a very important consideration in amplifier selection for final deployments this project. Typical power amplifier (PA) efficiency is on the order of 25%, although some more recent designs claim efficiencies of 40% or better.

Any specific recommendations on PA models are welcome.

Pre-Amplifiers

Matched to a 50 Ohm input, the USRP has a full-scale input power of 10 dBm. According to available documentation, the noise level is about 5% of full scale when the receiver is operated at maximum gain, giving a receiver noise floor of -16 dBm at the A/D converter following a gain of 90 dB. This corresponds to an input noise floor of -106 dBm at the input connector.

The thermal noise floor of the GSM 271 kHz channel is -119 dBm at 322K (120F). For the receiver to be thermal noise limited (instead of gain-limited), we need $-106 - (-119) = 13$ dB of external receiver gain between the USRP and the antenna, not counting additional gain required to compensate for cable and insertion losses.

We would welcome any specific preamplifier recommendations.

[8] We've had hot-shot radio guys tell us they can build one from a couple of empty paint cans and some copper foil.

3.8.3 Packaging and Power Supplies

We will eventually need field-ready packaging and a battery-backed renewable power system for this equipment, but have no specific plans as of yet.

Chapter 4

Project Plan and Milestones

We will build this BTS from the "bottom" up, starting at the physical layer and verifying performance with loopback tests in each layer.

4.1 Major Milestones

1. Transceiver Loopback. We send normal bursts populated with random bits through the transmitter, have the radio signal travel through a cable to the receiver, decode the received busts and verify that the transmitted and received bits match. For the first cut, this is a single-ARFCN GMSK implementation.

2. L1 FEC Loopback. We have L1 FEC encoders and decoders connected by a UDP fixture that simulates the transceiver.

3. Total L1 Loopback. We run the FEC test of Step 2 but replace the UDP test fixture with the actual transceiver and a cable loopback of Step 1.

4. Beacon Test. Generate a beacon and see if a handset responds with RACH bursts. This requires us to

   - generate System Information Messages 1-4 on the BCCH,
   - generate a correct SCH and FCCH,
   - generate empty paging messages for filling on the CCCH, and
   - generate dummy bursts on all other C0 timeslots.

   This test can also be verified with a beacon analyzer like the R&S CMD57 (Section 5.2.2).

5. L2 Loopback. We have a pair of L2 LAPDm processors "chatter" at each other with simulated dropped frames.

6. L1-L2 Loopback. We run the L2 pair from Step 5, but run the data path through the full L1 loopback of Step 3.

7. L3 "Headless". We run a full 3-layer GSM stack, but short-circuit a lot of control layer functions with hard-coded values and loopbacks. For example, the location updating operation will simple accept registrations without actually verifying IDs and the call connection functions will simply loop back vocoder frames. This is essentially the full GSM stack, but without any interactions with the outside world.

8. SIP Testing. We have a set of model SIP state machines that run registration and call setup transactions with the Asterisk server. Transactions will be tested in the same order in which they are listed in Section 3.7 using both local SIP softphone clients and external VoIP carriers.

9. Full L3. We replace all of the loopbacks and hardcoding of Step 7 with the real SIP transactions of Step 8. Transactions will be tested in the same order in which they are listed in Section 3.7, using local SIP softphone clients.

10. Real Telephone Service. We connect the Asterisk server to a VoIP carrier and start making real telephone calls.

4.2 Next Steps

Once the basic GSM L1-L3 stack is working, we can start adding features:

1. TMSI support, to allow partial anonymization of subscribers.

2. SMS text messaging.

3. Handovers.

4. Half-rate services.

5. Other vocoders (EFR, HR, AMR).

6. Multiple ARFCNs.

7. Frequency Hopping. Frequency hopping can actually simplify network planning. We will implement frequency hopping by inserting an extra sublayer between the transceiver interface and the TDM sublayer.

8. GPRS for packet switched data at rates up to about 100 kbit/s.

9. EDGE for packet switched data at rates up to about 250 kbit/s.

Unfortunately, we probably cannot legally support authentication and encryption in an open source system because the algorithms used in GSM are distributed only under non-disclosure agreements. While there have been unauthorized disclosures of these algorithms, we do not want to taint this project with material that would jeopardize its distribution status. Certainly, any private development project is free to integrate these features into controlled distributions given proper agreements with ETSI.

Chapter 5

Testing with Hardware

We want to provide a lot of good software unit-testing tools, but in the end you can only verify the operation of a BTS by testing it with real phones. To do this you need real equipment and a real testing environment.

5.1 RF Testing Environments

The absolute last thing you want to do is run a BTS in your local cellular or DCS/PCS band while announcing a local carrier's network identity parameters on your beacon. *This is no joke.* You will be interfering with the service of the licensed network of a common carrier.[1] Wherever you live this probably violates a number of civil and criminal codes. You will also be doing it in a very ham-fisted and obvious way. Running an IMSI-catcher without leaving a lot of evidence is an art and not one we intend to address in this project. Don't try it at home.

So given that, how do you test a BTS legally?

5.1.1 Closed RF Environments

The safest and surest way to test the OpenBTS is in a closed RF environment, with all signals confined in cables or Faraday cages. Some GSM phones and modems are connectorized in a way that affords this kind of testing. The MultiTech GSM/GPRS modems are a good example.

The advantage of the closed environment is that it is absolutely controlled, with known delays, known power losses, and known equipment present. The disadvantage is that it is expensive to wire up a lot of phones this way and even more expensive to build realistic propagation models.

[1] Worse yet, it's possible that an unsuspecting subscriber camps on your BTS and then tries to place an emergency call that you can't connect.

5.1.2 ISM Bands

In many parts of the world you can transmit up to 1 W (30 dBm) in certain unlicensed bands. Since the maximum power of a low-end GSM handset is 0.8 W there is the potential to build a GSM-based telephone system with a range of a few hundred meters at these power levels, even with small antennas. *This is a general statement, not legal advice. Know the specific regulations in your area.*

In North America there is a 900 MHz ISM band that overlaps with the downlink portion of the primary E-GSM 900 band, allowing a BTS transmitter to operate in the ISM band according to ISM transmission limits. Remember, though, that the corresponding E-GSM 900 uplink frequencies fall in the GSM 850 downlink band, so this is not a way to run an unlicensed cellular network. However, if you operate in a rural area with poor GSM coverage and far from major airports you can probably get an experimental license that would allow you to run a full-power (20 W) open-air test range in the ISM band.[2]

5.1.3 Limited Power

Depending on radio regulations in your area, you may be able to operate the BTS and handsets over a range of a meter or two at very low power levels. In order to do this correctly, you must limit the power not just of the BTS but also of the handset. Handset transmit power is controlled by the BTS, so this is possible. The problem, though, is that the "stock" USRP receiver is not sensitive enough to receive the uplink signal unless the handset transmit power is set very high. So, in order to do true limited-power testing, you will need a preamp on the input of the USRP, which adds several hundred dollars to the price of your development kit. This approach is risky, though, because if you do it wrong your first indication might be a visit from the authorities.

5.1.4 Fallow Spectrum

If you are a rural resident, chances are there are plenty of open ARFNs in your area, especially if cellular coverage is poor. Even though these ARFCNs are almost certainly licensed to a carrier, you may be able to get a temporary or experimental license to operate in this "fallow" spectrum under the condition that you do not interfere with licensed networks.[3]

5.2 Support Hardware

5.2.1 Reference Phones

You can do a lot of good testing with the right kind of unlocked phone.

[2] We have been granted such STAs (Special Temporary Authority).
[3] We have been granted this type of STA also, in rural Northern California.

"Field Test" Phones

Nokia phones using software versions DCT-3 and earlier can be placed in a *field test mode*. In this mode they report a wealth of information on their status, BTS parameters and radio performance. Of particular value, the phone will report the camped ARFCN and the channel type that is currently active.[4] On some older Nokia models, this feature can be enabled by entering `*3001#12345#` through keypad. In lated models, you will need a cable or IRDa link to enable this feature.

Some Motorola phones support a "net monitor" mode similar to the Nokia field test mode. In fact, many manufacturer provide similar hidden modes, especially in older models.

"Open" GSM Phones

Most GSM handsets are build around a core GSM module that contains all of the radio and telephone functions, controlled through an AT-type command interface based on the GSM 07.xx series of specifications. There are a few GSM phones out there that allow direct access to that interface. You can query these phones through a serial or USB port to get access to radio configuration and performance data. We have had good results with the MultiTech GSM modems. The FIC Neo1973 and TuxPhone might also fall into this class as useful test phones. Phones in this class are normally sold as unlocked. Some GPRS modem cards also offer this type of interface through a pseudo-serial device.

Non-Roaming SIMs

If you broadcast beacon parameters for a carrier that does not have roaming agreements with the carriers in your area, most phones in your area will not attempt to camp to your BTS. If your test phone uses a SIM for a carrier that does not have roaming agreements with the carriers in your area, your test phone will not attempt to camp to the local carriers' networks as long as it can find your BTS first. Given these conditions, non-local, non-roaming SIMs are very handy for testing and for minimizing the chance of accidental interference with your local carriers.[5] Of course, for non-roaming SIMs to be useful, your phone must be unlocked.

If you have a contact in the UK or travel there often, you can get SIMs from Tesco Mobile for about US$5 each at supermarkets all over the country. Tesco advertises UK-only service and to our knowledge Tesco Mobile has no roaming agreements outside of the UK. These SIMs are known to be incompatible with the AT&T and T-Mobile networks in the US. They expire after six months if they are not registered, but can still be used for testing with the understanding that the associated E.164 address (the telephone number) has been reassigned to another Tesco subscriber.

We'd be interested to hear about other SIM compatibility experiences.

[4] Some people are ignorant enough to think that this is a trade secret. That's pretty funny until they sue you for misappropriating their supposed IP.

[5] Again, some people are ignorant enough to think that this is a trade secret and arrogant enough to think they invented it.

5.2.2 Test Equipment

If you have deeper pockets you might save time and trouble with professional test equipment.

Spectrum Analyzers and Signal Generators

If you are working in the radio layer, these should be of obvious utility. The GSM channel bandwidth is just under 271 kHz. The standard GSM bands are 850 MHz, 900 MHz, 1800 MHz and 1900 MHz. If you are patient, you might put together a spectrum analyzer from Tektronix 7000-series mainframe components on eBay for US$1k-US$2k.[6] If anyone has a favorite low-cost GMSK generator, other than the USRP itself, please let us all know about.

Rohde & Schwarz BTS Verification Sets

The R&S CMU200 and CMU300 are BTS verification test sets. They make radio measurements (phase noise, frequency drift, power ramping, etc.), decode the SCH and BCCH and run common transactions with protocol logging. They also provide test modes related to the Abis interface that are not of value in this project. These boxes sell for US$20k to US$30k used, depending on installed options.

The R&S CMD57 offers many of the features of CMU series test sets but at a much lower price. The CMD57 has been discontinued but is available used for US$4k-US$6.[7]

Sagem Network Engineering Phones

The Sagem OT-series test phones look like ordinary cell phones but are sold as test equipment. These devices can be forced to camp on specific ARFCNs and decode and log L2 and L3 transactions on all control channels, including CCCH transactions for co-cell subscribers. The various models sell for US$2k to US$5k new depending on the bands and service types they support. They also require a Windows PC to run the control and logging software.

[6] We have a Tek 7603 frame with a 7L12 plugin. The 7L12 has option 39, going up to 2.5 GHz. Special thanks to the friend who donated that 7603. You know who you are.

[7] We bought one of these from a used equipment dealer in the UK for about US$5k.