Integration of Open-Source GSM Networks

Thomas A. Cooper

Thesis submitted to the faculty of the
Virginia Polytechnic Institute and State University
in partial fulfillment of the requirements for the degree of

Master of Science
in
Electrical Engineering

April 27, 2012
Blacksburg, Virginia

Keywords: Open-Source Software, Network Architecture, Wireless, Software-Defined Radio, Cellular Network, GSM, Base Transceiver Station, Protocol Layers

Integration of Open-Source GSM Networks

Thomas A. Cooper

ABSTRACT

Global System for Mobile Communications (GSM) networks are receiving increasing attention in the open-source community. Open-source software allows for deployment of a mobile cellular network with lower costs, more customization, and scalable control. Two popular projects have emerged that offer varying network architectures and allow users to implement a GSM network in different capacities depending on individual needs. Osmocom provides more network control and scalability but requires commercial Base Transceiver Station (BTS) hardware with limited availability and closed source code. OpenBTS provides minimal GSM network functionality with more easily available and open-source hardware; however, it does not allow multi-cellular network configuration.

This thesis offers a significant contribution towards a fully open-source GSM network by integrating the two major open-source communities, Osmocom and OpenBTS. Specifically, the Osmo-USRP program provides an inter-layer interface between the different network architectures of two GSM base station projects. Inter-layer primitive messages are processed in a thread multiplexer that manages logical channels across the interface. Downstream flow control is implemented in order to receive data frames on time for transmitting at the appropriate GSM frame number (FN). Uplink measurements, which are necessary for decision making in the Base Station Controller (BSC), are also gathered in the physical layer of Osmo-USRP and reported to Osmocom.

Osmo-USRP operation is tested using a Universal Software Radio Peripheral (USRP), a relatively inexpensive and accessible Software-Defined Radio (SDR). Standard GSM events are investigated for single cell and multi-cellular network configurations. These tests include subscriber authentication and encryption, location updating, International Mobile Subscriber Identity (IMSI) attach and detach, Short Message Service (SMS) storage and delivery, voice calls with the full-rate audio codec, and uplink and downlink measurement reporting. While most functionality is successfully tested, inter-cell handover is not currently implemented. Further details on the proposed implementation of program limitations, especially inter-cell handover, are also discussed.

This work is supported by the U.S. Army Communications-Electronics Research, Development and Engineering Center (CERDEC).

Acknowledgments

I would like to thank Dr. Reed and Dr. Tim Newman for supporting me in undergraduate research and convincing me to continue my education into graduate school.

I would also like to thank my other committee members, Dr. Clancy and Dr. Bose, and the always helpful students in the MPRG labs for their constant support and encouragement.

Finally, I would like to thank Tom Tsou for his continuous guidance and allowing me to run most of my ideas past him.

Contents

List of Figures

List of Tables

List of Abbreviations

3GPP	3rd Generation Partnership Project
AB	Access Burst
AGCH	Access Grant Channel
ARFCN	Absolute Radio Frequency Channel Number
AuC	Authentication Center
BB	Baseband
BCCH	Broadcast CCH
BCH	Broadcast Channel
BSC	Base Station Controller
BSIC	Base Station Identity Code
BSS	Base Station Subsystem
BTS	Base Transceiver Station
CBCH	Cell Broadcast Channel
CC	Call Control
CCCH	Common CCH
CCH	Control Channel
CDMA	Code Division Multiple Access
CM	Connection Management
DCCH	Dedicated CCH
EIR	Equipment Identity Register
FACCH	Fast Associated CCH
FB	Frequency Correction Burst
FCCH	Frequency Correction Channel
FDMA	Frequency Division Multiple Access
FEC	Forward Error Correction
FER	Frame Error Rate
FIFO	First In First Out
FN	Frame Number
GGSN	Gateway GPRS Support Node
GMSC	Gateway MSC
GPRS	General Packet Radio Service
GSM	Global System for Mobile Communications
GSMA	GSM Association
GTP	GPRS Tunneling Protocol
HLR	Home Location Register
IMEI	International Mobile Equipment Identity
IMSI	International Mobile Subscriber Identity
IP	Intellectual Property, Internet Protocol
ISDN	Integrated Services Digital Network
L#	Layer #
LAPDm	Link Access Procedure on Dm Channel
LCS	Location Services

ME	Mobile Equipment
MM	Mobility Management
MO	Mobile Originated
MS	Mobile Station
MSC	Mobile Switching Center
MT	Mobile Terminated
MTP	Message Transfer Part
NB	Normal Burst
NCH	Notification Channel
NITB	Network in the Box
NSS	Network Switching Subsystem
OML	Operation and Maintenance Link
PBX	Public Branch Exchange
PCH	Paging Channel
PSTN	Public Switched Telephone Network
RA	Random Access
RACH	RA Channel
RR	Radio Resource Management
RSL	Radio Signaling Link
RSSI	Received Signal Strength Indication
SABM	Set Asynchronous Balanced Mode
SACCH	Slow Associated CCH
SAP	Service Access Point
SAPI	SAP Identifier
SB	Synchronization Burst
SCCP	Signaling Connection Control Part
SCH	Synchronization Channel
SDCCH	Stand-alone DCCH
SDR	Software-Defined Radio
SGSN	Serving GPRS Support Node
SI	System Information
SIM	Subscriber Identity Module
SIP	Session Initiation Protocol
SMS	Short Message Service
SS	Supplementary Services
SS7	Signaling System 7
TA	Timing Advance
TCH	Traffic Channel
TDMA	Time Division Multiple Access
TMSI	Temporary Mobile Subscriber Identity
TS	Timeslot
UA	Unnumbered Acknowledge
UHD	USRP Hardware Driver
USRP	Universal Software Radio Peripheral
VLR	Visitor Location Register
VoIP	Voice over Internet Protocol

1. Introduction

The Global System for Mobile Communications (GSM) network is a budding area of research in the open-source community. This technology is more favorable to developers than its Code Division Multiple Access (CDMA) counterpart because of several reasons: The GSM specifications are publicly available through the 3rd Generation Partnership Project (3GPP) and not protected Intellectual Property (IP); the capacity and bandwidth are more scalable for small cells; and the physical layer can be deployed via an inexpensive Software-Defined Radio (SDR) [1].

GSM is also the most widespread mobile technology with over 5 billion connections in 2010 according to the GSM Association (GSMA) [2]. Although Europe is the most heavily penetrated region, GSM is the only option in many developing and conflicted areas of Africa and Asia, in which growth is highest. Open-source software allows for deployment of a mobile network with lower costs and more customization and control. Originally intended to promote research and experimentation, the role of open-source systems as competitors to traditional GSM network providers has increased. Companies such as Range Networks and Sysmocom GmbH have brought these custom deployments to realization by using cheaper hardware and software than other commercial GSM networks. These small-scale networks are operated at about a tenth of the cost of traditional networks by using SDRs and Voice over Internet Protocol (VoIP) backhaul, according to Range Networks [3].

This work, the creation of an open-source software program named Osmo-USRP, offers a significant contribution towards a fully open GSM network by integrating the two major open-source projects, OpenBTS and Osmocom. Osmo-USRP allows Osmocom developers to use cheap and available SDR hardware, much like the many OpenBTS users. OpenBTS advocates can also use this program to interface with standard GSM network components and extend control over multiple cells in their networks; the primary capability enabled by this work is the multi-cellular network configuration. Multiple Osmo-USRP processes can be controlled simultaneously via the Virtual Teletype Terminal (VTY) command interface of an OpenBSC process. From a centralized access point, OpenBSC manages a subscriber database, neighboring cell arrangements, and network functions such as location updating; inter-cell handover will also be supported in future work. For example, a Mobile Station (MS) can change its current serving cell based on the measured signals of neighboring cells; when also moving between location areas, the location update procedure is undergone to update the corresponding Visitor Location Registers (VLR). Specifically, Osmo-USRP allows multi-cellular functionality and configuration among cells using the Universal Software Radio Peripheral (USRP) SDR platform. Osmo-USRP provides a foundation for open-source developers from both communities to expand this work through their own contributions. OpenBTS-Osmo serves as a foundation for this work; like Osmo-USRP, the project is intended to connect OsmoBTS and OpenBTS but is unfinished in its current state. Some part of Osmo-USRP is expected to be eventually integrated into the Osmocom project’s repository. Both Osmocom software and the public release of OpenBTS are compatibly licensed under the GNU Affero General Public License, version 3 (AGPLv3); therefore, there are no significant issues with merging code from the two projects into Osmo-USRP.

In order to test the operation of Osmo-USRP, standard GSM events are investigated for single cell and multi-cellular network configurations. These tests include subscriber authentication and encryption, location updating, International Mobile Subscriber Identity

(IMSI) attach and detach, Short Message Service (SMS) storage and delivery, mobile originated (MO) and mobile terminated (MT) voice calls with the full-rate audio codec, and uplink and downlink measurement reporting. Most of the standard functionality is successfully tested, but the inter-cell handover procedure is not currently implemented. This work provides a discussion of the proposed handover implementation for future work.

Section 2 provides extensive background on the GSM network and its components, protocol layers, and functions; the publicly available open-source projects and their network configurations are also described here. Section 3 outlines the objectives of this work and the approaches required to fulfill them. The design and implementation of the Osmo-USRP interface program are found in Section 4. Section 5 covers the results of testing Osmo-USRP in an open-source GSM network of single and multiple cells, as well as concerns with inter-cell handover. Section 6 discusses the limitations in the current version of the work along with recommendations for addressing these in future work. Finally, concluding remarks on this work are found in Section 7.

2. Background

This section describes the network architecture, components, and protocol layers of the GSM technical specification. Current open-source projects and their available network configurations are also provided.

2.1 GSM Overview

A GSM cellular network functions to maintain a continuous connection for mobile users. Figure 2.1 shows the hierarchical structure of the network. Each cell contains a Base Transceiver Station (BTS) that provides an Um air interface to multiple MSs. The regional functionality and interconnectivity of multiple BTSs is controlled by a Base Station Controller (BSC). Routing and switching controls between multiple BSCs, as well as the Authentication Center (AuC) and VLR, Home Location Register (HLR), and Equipment Identity Register (EIR) databases, are managed by a Mobile Switching Center (MSC). Section 2.2 describes each of these major components of the GSM network structure in more detail.

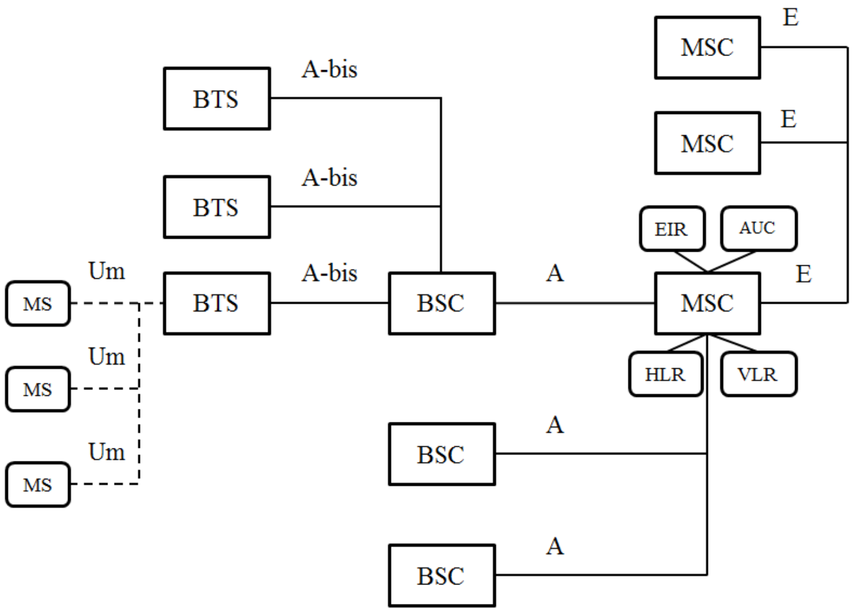

Figure 2.1: Hierarchical structure of a GSM network.

The GSM protocol stack in Figure 2.2 contains three primary layers that provide messaging communication between different components of the network system; the highlighted layer blocks are studied in this work. Layer 1 (L1) contains the physical Time Division Multiple Access (TDMA) channels of the Um interface between BTS and MSs. In Layer 2 (L2) of Um, the data link layer contains Link Access Procedure on Dm Channel (LAPDm), a GSM version of LAPD from ISDN. Layer 3 (L3) is composed of several management sub-layers; on the network side, Radio Resource (RR) management resides mostly in the BSC with some functionality in the BTS and MSC, and Mobility Management (MM) and Connection Management (CM) exist only in the MSC. RR manages the allocation, configuration, and connection of radio channels. MM handles mobility-related functions of the MS, such as location updating, authentication, and security. CM is concerned with managing Call Control (CC) and other service entities, such as Short Message Service (SMS) [4]. The Radio Signaling Link (RSL) of the A-bis interface carries RR messages between BTS and BSC, as well as transparent MM and CM messages between MS and MSC; additionally, the Operation and Maintenance Link (OML) transports messages for network management and BTS monitoring. On the A interface, Message Transfer Part (MTP) of Signaling System 7 (SS7) is used in L1-3. Section 2.3 provides more information about these important pieces of the GSM protocol stack.

This work directly involves L1-L2 inter-layer communication within the BTS; therefore, the L1 logical channels described in Section 2.3.2, as well as TDMA structure on the Um interface, contain the most relevant background information. Some RR functions such as channel activation are also important to managing the BTS L1; however, higher layer functions, such as the L3 RR, MM, and CM procedures, are mostly transparent to L1-L2 of the BTS. These L3 messages must still be analyzed when studying and testing standard GSM events.

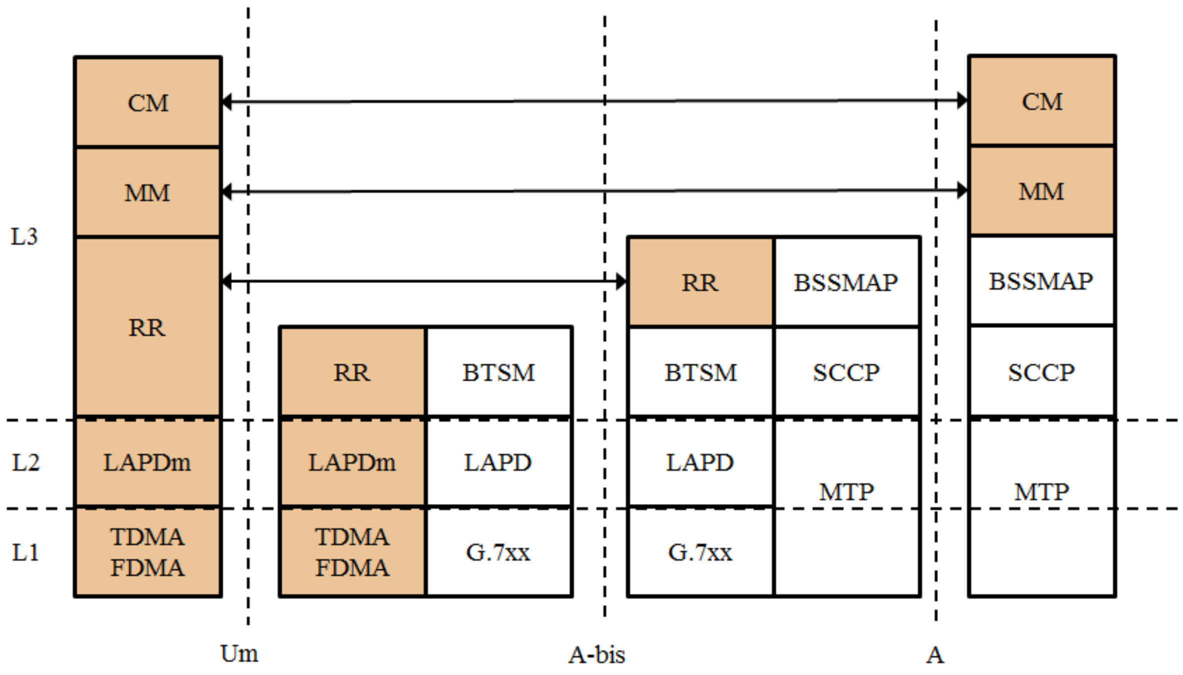

Figure 2.2: Layers of the GSM protocol stack.

2.2 Network Components

A traditional GSM network is composed of several hierarchical entities, which are described in this section at a level necessary for basic understanding.

2.2.1 MS/SIM

The MS is a user handset that connects to the BTS over the Um air interface. It consists of Mobile Equipment (ME) that contains a unique International Mobile Equipment Identity (IMEI) and a Subscriber Identity Module (SIM) card that stores a unique International Mobile Subscriber Identity (IMSI) and authentication key. These secure numbers are also stored in the network and used to properly identify and authorize the subscriber. Also of note, the MS does not authenticate the network to which it connects, enabling malicious attacks by falsifying a man-in-the-middle BTS.

2.2.2 BTS

The BTS provides the Um radio interface for MSs within its serving cell. A BTS can contain multiple transceivers for transmitting and receiving on multiple frequencies and sectors in one cell. The actual management and functionality is controlled by the BSC over the A-bis interface. A network of BTSs and the governing BSC makes up the Base Station Subsystem (BSS).

2.2.3 BSC

The BSC manages the resources, services, and networking of multiple BTSs. Radio frequencies are mapped into logical channels, and dedicated channels are allocated as needed for signaling and traffic connections. A MS sends location update requests to the BSC to allow the network to track it. The BSC also collects periodic measurement reports from a connected MS in order to decide if handover to a better quality cell (inter-BTS) or frequency (intra-BTS) is required, including during a call. Further elaboration on procedures of RR, MM, and CM is found in Sections 2.3.3-5.

2.2.4 MSC/GMSC

The MSC connects to a BSC via A interface and to another MSC via E interface. A single MSC provides regional network management to multiple BSCs, including functions such as call routing, MS authentication and location tracking, inter-BSC handover, and SMS delivery. If the target MS is not located on the original MSC, the Gateway MSC (GMSC) is responsible for locating which MSC contains the target MS. The GMSC also connects the mobile network to the Public Switched Telephone Network (PSTN) and therefore other mobile networks. The MSC and its VLR, HLR, EIR, and AuC make up the Network Switching Subsystem (NSS).

2.2.5 HLR/VLR/EIR

The HLR is a central database that stores the unique IMSI of each MS subscriber on the network, as well as other subscriber data. This database is accessed in order to update a VLR database when the MS roams to another MSC. The VLR also informs the HLR of the new location of an MS. While the MS stays within the network of the governing MSC, the VLR is used for authentication and location updating. Together, the original HLR and current VLR help the MSC to find the specific cell containing an MS anywhere in the mobile network.

The EIR is another database that contains IMEIs of handsets that are identified as stolen and can be tracked.

2.2.6 AuC

The AuC uses an encrypted procedure to verify that the IMSI and authentication key stored in a SIM card matches those authorized by the network. If a connected MS is authenticated, the AuC also provides the appropriate BSC with an encryption key to protect further communication with the MS.

2.2.7 GGSN/SGSN

When the network supports General Packet Radio Service (GPRS), a Gateway GPRS Support Node (GGSN) is used to connect to other packet switched networks, such as the Internet. The GGSN routes incoming data to the Serving GPRS Support Node (SGSN) that contains the BSS of the target MS. The SGSN handles mobility, connection, and authentication management in its network area, as well as storing GPRS users and their locations in a database. The GGSN also formats and routes outgoing GPRS data to external packet data networks.

2.3 Protocol Layers

The layers and interfaces of interest in GSM protocol stack are examined in this section. These include L1 and L2 of Um interface and the resource, mobility, and connection management functions in L3. GPRS-specific mobility and session management functions in L3 are not discussed.

2.3.1 L1: FDMA/TDMA

GSM uses Frequency Division Multiple Access (FDMA) to divide larger spectrum bands into smaller 200 kHz channels. Each pair of downlink and uplink channels in a given GSM band is assigned an Absolute Radio Frequency Channel Number (ARFCN) in order to identify the specific frequencies with a standard integer number.

In addition to FDMA, GSM uses TDMA to further divide each frequency channel into eight timeslots (TS), allowing eight users to share a single frequency. Each timeslot is 576.9 microseconds in length, and a TDMA frame consisting of eight timeslots is 4.615 milliseconds in length [5]. A Control Channel (CCH) multiframe contains 51 TDMA frames, while a Traffic Channel (TCH) multiframe contains 26 TDMA frames. A CCH superframe contains 26 CCH multiframes and a TCH superframe contains 51 TCH multiframes; therefore both types of superframe contain 1326 TDMA frames. The Frame Number (FN) restarts every hyperframe, or 2048 superframes.

2.3.2 L1: Logical Channels

Certain TDMA frames in each timeslot are organized into logical channels, adhering to the specific configuration of a BTS [6]. This organization allows specific data to be addressed by TS and FN. The three categories of signaling channels are Broadcast Channel (BCH) for unidirectional broadcast functions, Common Control Channel (CCCH) for point-to-multipoint access functions, and Dedicated Control Channel (DCCH) for bidirectional point-to-point functions; there is also the TCH for carrying encoded speech and data. The different types of logical channels [7] are briefly described in Table 2.1.

Table 2.1: Overview of logical channel types and functionality.

Channel	**Type**	**Brief Description**
FCCH	BCH	Broadcasts unmodulated frequency correction bursts (all zeros) to synchronize the frequency of the MS.
SCH	BCH	Broadcasts Base Station Identity Code (BSIC) and FN to identify the BTS and synchronize FN of the MS.
BCCH	BCH	Broadcasts cell identification parameters, including network codes, logical channel configurations, frequencies of neighbor cells, and synchronization information.
CBCH	BCH	Optionally assigned to SDCCH; broadcasts network-defined messages such as public service announcements to all MSs; not used in this work.
RACH	CCCH	Receives random access (RA) bursts on uplink when the MS requests a SDCCH.
AGCH	CCCH	Broadcasts replies to grant access to a specific MS when assigning a SDCCH.
PCH	CCCH	Broadcasts paging requests to inform a specific MS of incoming traffic data.
NCH	CCCH	Broadcasts notifications to inform a specific MS of incoming group and broadcast calls; not used in this work.
SDCCH	DCCH	Carries signaling for functions between MS and BTS, such as location updating and call setup; carries SMS data when MS is not in call.
SACCH	DCCH	Always associated with SDCCH or TCH; carries signaling for power control, synchronization, and channel measurements, as well as MS power and timing measurements; carries SMS data when MS is in call.
FACCH	DCCH	Steals frames from TCH (at expense of data rate) instead of establishing stand-alone channel; carries signaling for call setup and handover control.
TCH	TCH	Carries encoded speech and data for single direction, at either full rate (24 frames of 26 in multiframe) or half rate (12 frames).

Logical channels are grouped into the predefined combinations shown in Table 2.2, which repeat every multiframe [5], [7].

Table 2.2: Composition of predefined logical channel combinations (C1-C10).

	C1	C2	C3	C4	C5	C6	C7	C8	C9	C10
TCH/F	1							1	1	1
TCH/H		1	2							
FACCH/F	1							1		
FACCH/H		1	1							
SACCH	1	1	1		4		8	1	1	1
SDCCH					4		8			
FCCH				1	1					
SCH				1	1					
BCCH				1	1	1				
CCCH				1	1	1				

At least one combination with BCHs (e.g. C5) is required, and the remaining timeslots can be filled with DCCH (e.g. C7) and TCH (e.g. C1) combinations as dictated by the intended cell capacity. Only BCHs and CCCHs are allowed in the first ARFCN. Some examples of common combinations, C1, C5, and C7, are illustrated in Figures 2.3-2.11 [8]; these three combinations are used in the timeslot configuration of this work. In C5 and C7, there are two alternating multiframe configurations; when a Slow Associated CCH (SACCH) is associated with a Stand-alone Dedicated CCH (SDCCH), it only requires half as many frames. There is also an offset between downlink and uplink of 15 frames, which allows the MS to receive on a certain logical channel and transmit a response in 15 frames instead of waiting an extra multiframe. Note that the uplink multiframe for C1 is the same as Figure 2.3 except for this 15-frame shift.

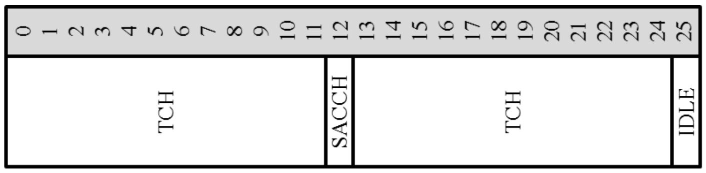

Figure 2.3: Multiframe configuration for downlink of C1.

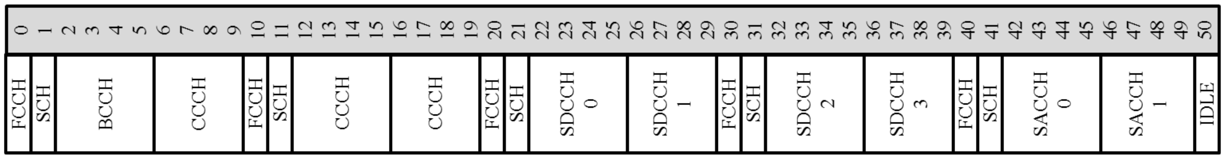

Figure 2.4: Multiframe configuration (#1) for downlink of C5.

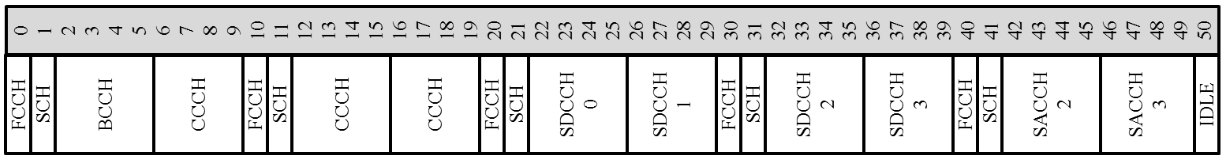

Figure 2.5: Multiframe configuration (#2) for downlink of C5.

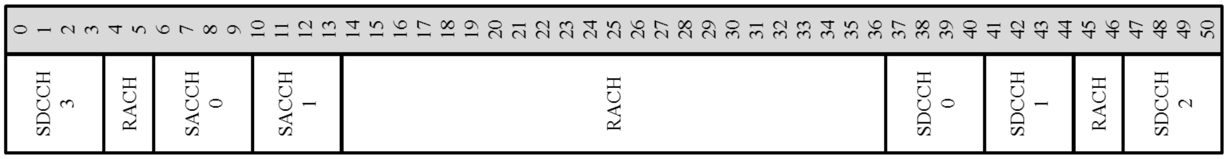

Figure 2.6: Multiframe configuration (#1) for uplink of C5.

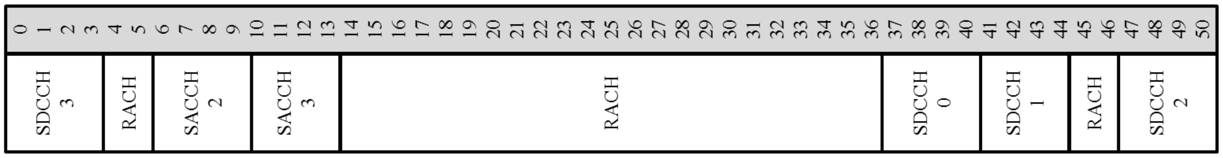

Figure 2.7: Multiframe configuration (#2) for uplink of C5.

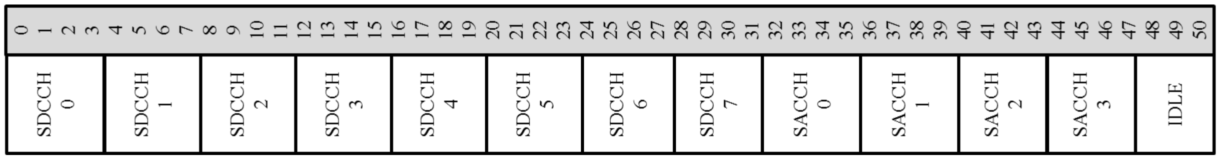

Figure 2.8: Multiframe configuration (#1) for downlink of C7.

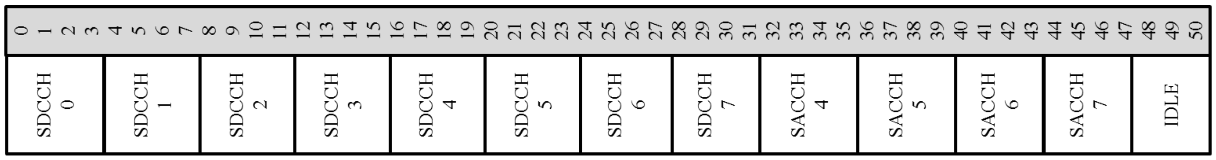

Figure 2.9: Multiframe configuration (#2) for downlink of C7.

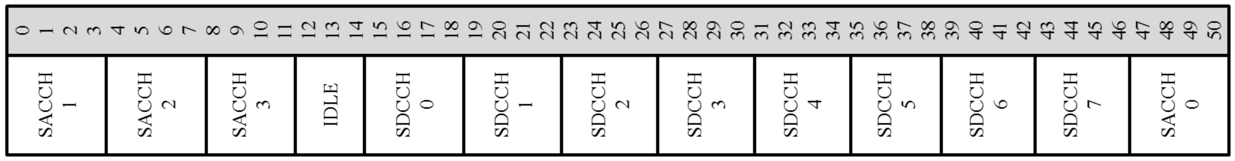

Figure 2.10: Multiframe configuration (#1) for uplink of C7.

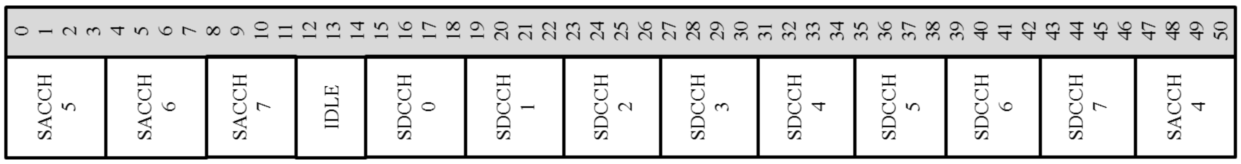

Figure 2.11: Multiframe configuration (#2) for uplink of C7.

2.3.3 L2: LAPDm

L2 of the Um interface uses LAPDm, a modified version of LAPD from ISDN. LAPDm accounts for uncertainty in wireless data link establishment by implementing contention resolution [9]. This process is required when two MSs attempt to establish a SDCCH using identical RA bursts, therefore the first L3 frame is echoed back to the MS to verify that it has rightful access of the data link. LAPDm also only supports two Service Access Point Identifiers (SAPIs) for connecting to L3 services: 0 for RR, MM, and CC and 3 for SMS.

2.3.4 L3: RR

The Radio Resource (RR) management sublayer is the lowest sublayer of L3, providing services to the MM sublayer [10]. These services include establishing control and traffic channel connections, ciphering mode indication, releasing control channel connections, and control data transfer. The overall purpose of RR procedures is to establish, maintain, and release RR connections in order to manage physical channels and control data links. An RR connection in dedicated mode performs procedures such as channel assignment, handover, and channel mode modify.

2.3.5 L3: MM

The Mobility Management (MM) sublayer of L3 provides services to CM sublayer entities [10]. The general purpose of these services is to support the mobility and registration of MSs throughout the GSM network. MM procedures always require an RR connection between MS and the network; these procedures include authentication, identification, Temporary Mobile Subscriber Identity (TMSI) reallocation, IMSI attach and detach, location updating, and periodic updating.

2.3.6 L3: CM

The Connection Management (CM) sublayer of L3 consists of four main entities: Call Control (CC), Supplementary Services (SS), Short Message Service (SMS), and Location Services (LCS). CC services include mobile originated (MO) and mobile terminated (MT) normal call establishment, MO emergency call establishment, call maintenance and termination, and call related support of SS [10]. SS contains support for functions such as call forwarding, barring, holding, and waiting. SMS is used for sending point-to-point text messages along dedicated control channels (SDCCH or SACCH) [11]. LCS initiates and supports positioning measurements at the MS [12].

2.4 Open-Source Projects

There exist many active open-source projects in the area of GSM network components. The bulk of these projects are common libraries, tools, and GSM components managed by Osmocom developers. These components implement integrated and stand-alone features of network components, yet require specific BTS hardware. Several of the more relevant projects are described in Sections 2.4.1-5.

The other primary open-source project is OpenBTS, which implements minimal functionality of a GSM network to operate as a single femtocell. As the original creators of OpenBTS, Range Networks is responsible for developing the commercial closed source release and maintaining the public open-source release. For purposes of research and experimentation, only the open-source code is available and therefore further mention of OpenBTS, unless explicitly stated, is referring to this branch of the program. The related projects and hardware are described in Sections 2.4.6-8.

The two projects allow users to implement a GSM network in different capacities depending on individual needs. Osmocom provides more network control, functionality, and scalability but requires commercial BTS hardware with limited availability and higher cost. On the other hand, OpenBTS provides a stand-alone software implementation of a BTS with more easily available hardware but does not allow as much multi-cellular network functionality as Osmocom. Until this work, these two projects were not compatible with each other. Further motivations and objectives for this work are discussed in Section 3.

Figure 2.12 illustrates the components of the GSM network and portions of the GSM stack that each open-source project implements; the Osmo-USRP program, which interfaces OsmoBTS software and USRP hardware, is highlighted.

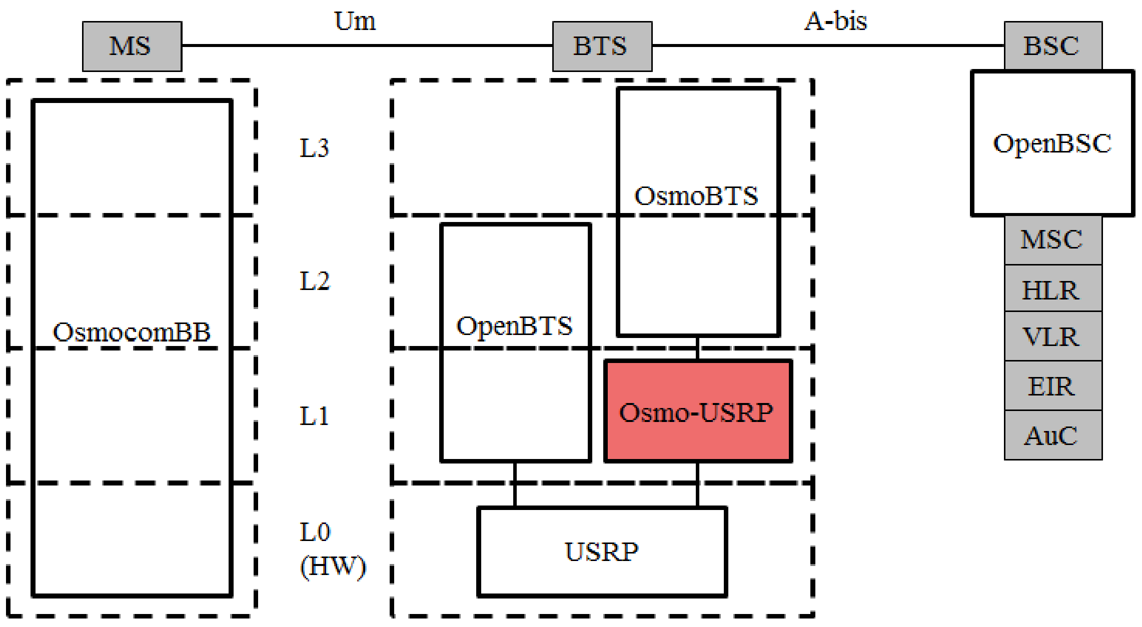

Figure 2.12: GSM implementations of open-source projects.

2.4.1 OpenBSC

OpenBSC is the primary project group for GSM-related components in Osmocom [13]. It consists of many software utilities and stand-alone programs tailored for specific GSM configurations. There are two separate main BSC applications within the project, Osmo-NITB and Osmo-BSC; these programs must be built with Libosmocore and Libosmo-Abis libraries. Osmo-NITB (Network in the Box) is a stand-alone GSM network that combines functionality from a BSC, MSC, HLR, VLR, EIR, and AuC into one package. The A-bis interface connects via IP or E1 to one of three commercial BTSs: Siemens BS11 microBTS, ip.access nanoBTS, or Ericsson RBS 2000 series; there are also plans to use OsmoBTS to support the upcoming SysmoBTS hardware by Sysmocom GmbH and even an experimental OsmocomBB phone setup. Using Osmo-NITB, it is possible to configure and control multiple BTSs via terminal commands without connecting to GSM core network components such as an MSC.

Osmo-BSC functions solely as a BSC to connect to an IP GSM network, with the same A-bis interface types but also with an A interface using Signaling Connection Control Part (SCCP) over IP. Therefore it also requires the Libosmo-SCCP library to be built.

2.4.2 OsmoBTS/OpenBTS-Osmo

OsmoBTS is an incomplete BTS for use with different hardware L1 implementations [14]; it contains A-bis over IP interface, L3, L2, and hooks for L1 primitives. OsmoBTS is primarily targeted for the upcoming SysmoBTS hardware by Sysmocom GmbH, although it also supports an experimental OsmocomBB phone setup. This project is still under heavy development and lacks some functionality such as handover procedure.

The OpenBTS-Osmo project [15] aims to connect OpenBTS L1 to OsmoBTS in order for Osmocom to support USRP hardware. However, this incomplete program is still in the early stages of a slow development process; OpenBTS-Osmo provides a foundational structure for the work presented.

2.4.3 OsmocomBB

OsmocomBB (Baseband) is a full MS GSM protocol stack that mirrors L2-L3 in OsmoBTS, along with L1 firmware [16]. Hardware drivers support the Texas Instruments Calypso/Iota/Rita GSM baseband chipset. Although there is only a command line UI, OsmocomBB allows a user to build and operate an entire MS with open-source software.

2.4.4 Libosmocore/Libosmo-Abis

The Libosmocore library [17] contains utility functions and GSM specifications that are deemed as common to multiple Osmocom projects, especially OpenBSC and OsmocomBB. Libosmo-Abis [18] contains common GSM functions specific to the A-bis interface, as well as drivers for A-bis over IP and E1 interfaces.

2.4.5 OsmoSGSN/OpenGGSN

OsmoSGSN is an implementation of the SGSN component used in a GPRS network [19]. The application connects to OpenBSC via the Gb interface and to OpenGGSN via GPRS Tunneling Protocol (GTP). OpenGGSN implements the GGSN component of the GPRS network [20]. Since GPRS is not supported by OpenBTS or OsmoBTS, these elements are not further discussed in this work.

2.4.6 USRP

The USRP is a relatively inexpensive SDR hardware platform built by Ettus Research [21]. The price and availability, as well as the wide selection of tools already developed for its use, make the USRP a valuable tool for anyone conducting research and experimentation. Considering the difficulty of acquiring a commercial BTS for use with Osmocom projects, the USRP serves as a major motivation for integrating the USRP support of OpenBTS with Osmocom.

The USRP is essentially an FPGA board with interchangeable RF daughterboards for transmitting and receiving in different frequency ranges. There are several USRP products designed by Ettus Research: the original USRP 1, B100, discontinued 2, N200, N210, E100, and

E110. These variations offer a range in prices, USB 2.0 vs. GigE host interface, host bandwidths, MIMO capability, and embedded CPU. Depending on the individual experimental setup and application, an external reference clock may also be necessary for clock stability.

2.4.7 GNU Radio/UHD

GNU Radio is an extensive library of SDR blocks and applications [22]. Its Libusrp sub-library is required to build OpenBTS when configured with a USRP 1. When using any other USRP, Ettus Research's USRP Hardware Driver (UHD) library is required with OpenBTS. More information on configuring OpenBTS for a specific USRP is found in the Appendices.

2.4.8 OpenBTS

The OpenBTS project [23] implements an independent BTS that interfaces with a Public Branch Exchange (PBX) via Session Initiation Protocol (SIP) in L3; the USRP is used to present the Um air interface in L1. By providing minimally integrated BSC and MSC functionality and using the PBX for L3 switching and calling functions, OpenBTS functions as a GSM access point that is more independent than a traditional BTS that requires a controlling BSC and core network. With this lack of a standard A-bis interface, OpenBTS does not connect to existing GSM networks or the Osmocom family of GSM components. The open-source version of this program does not allow the control or functionality of a BSC, including handover and multi-cellular parameter configuration. Asterisk or FreeSWITCH is used as the software PBX that handles private call routing and VoIP connection to the PSTN or Internet. Figure 2.13 [23] shows the architecture of the interconnecting layers and components of a standard OpenBTS network.

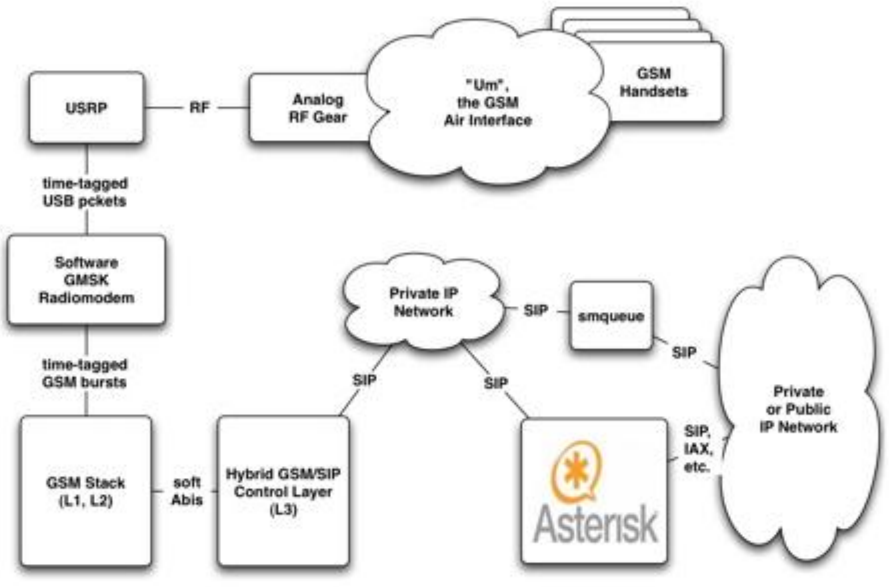

Figure 2.13: Hybrid GSM/SIP architecture of OpenBTS.
(Original online image retrieved from [23] on April 26, 2012.)

2.5 Available Network Configurations

There are two currently available options for running an open-source GSM network. One can attempt to acquire a commercial BTS and control it using OpenBSC, or one can purchase a relatively inexpensive USRP to use as the radio hardware for OpenBTS. The appropriate configuration choice depends on price as well as size and functionality of the user's desired GSM network.

2.5.1 Commercial BTS and OpenBSC

Using a commercial BTS with OpenBSC allows for the operation of a scalable multi-cellular network. However, commercial BTSs are much more expensive and difficult to buy than USRP family counterparts; they are also closed source unlike OpenBTS. Only a few models are supported by OpenBSC, and these BTSs are hard to acquire in single units for the average researcher or hobbyist. Sysmocom GmbH offers custom pre-configured GSM networks and services, including commercial BTSs; this method may be the easiest option if a user requires a commercial BTS.

OpenBSC also connects to OsmoBTS, allowing developers to hook into interchangeable L1 hardware. Sysmocom GmbH is planning on releasing a new SysmoBTS product in order to provide an affordable alternative to the commercial BTS.

This work also uses the OpenBTS-Osmo foundation to allow support for the USRP hardware in OsmoBTS as another alternative configuration.

2.5.2 USRP and OpenBTS

Using a USRP product with OpenBTS allows for the operation of a single access point with minimal GSM network components. While it is easy to deploy OpenBTS by connecting to a PBX, there is no multi-cellular control or functionality. Additionally, OpenBTS does not connect to legacy GSM core infrastructure as is possible with OpenBSC.

As previously stated, this work allows OpenBTS to hook into OsmoBTS, providing OpenBTS with a connection into the GSM core network via OpenBSC. This configuration also allows an open-source BTS component, composed of OpenBTS L1 and Transceiver code, for a more completely open-source GSM network.

3. Objectives

Integrating OpenBTS and OpenBSC connects two major open-source projects together, allowing for more control and customization of GSM components and protocols. Osmo-USRP is designed to provide such an inter-layer interface between OpenBTS L1, written in C++, and OsmoBTS L2, written in C. Figure 3.1 shows the highlighted position of Osmo-USRP in a GSM network constructed from open-source programs. The standard GSM network structure is also included for comparison.

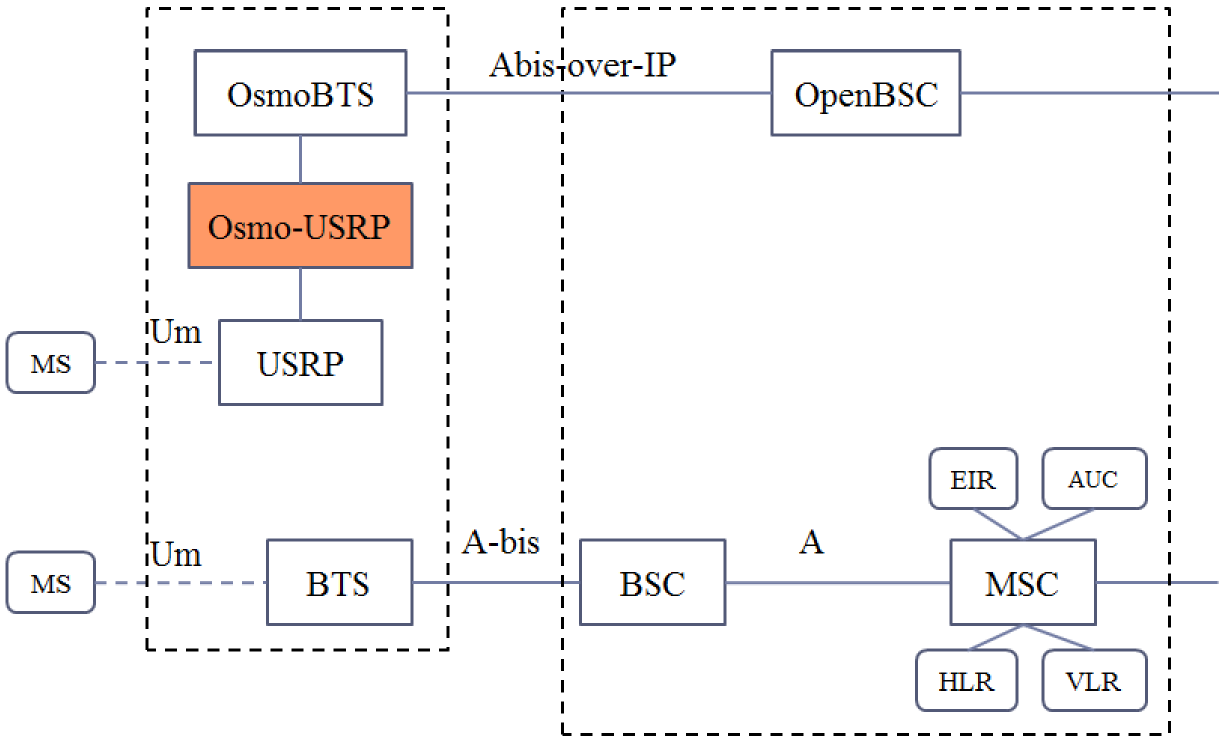

Figure 3.1: Comparison of GSM networks composed of open-source programs and traditional GSM components.

In order to ensure the successful operation and debugging of the Osmo-USRP program, many typical GSM events are tested for different network configurations. These events include authentication and encryption for an authorized subscriber; IMSI attach and detach; location and periodic updating; SMS storage, paging, and delivery via SDCCH; paging and MO and MT voice calling using the full-rate audio codec [24]; monitoring TCH, SDCCH, and idle channels to send uplink measurement reports; and sending downlink measurement reports via SACCH for serving and neighboring cells. Authorization, IMSI attach and detach, and location updating are necessary for the MS to initially access the network. SMS delivery and voice calling are desired practical operations for the MS. Uplink and downlink measurement reporting is necessary for the network to make informed decisions about cell selection, handover, RF power control, and determining radio link failure [25].

For a multi-cellular network, these standard GSM events are also tested across cells. Intra-BSC handover procedure [26] is also tested during a voice call for a MS moving into a new cell region.

Additionally, several software and hardware configurations are tested. For a single cell, OpenBSC and Osmo-USRP run on a single PC, or OpenBSC and Osmo-USRP run on separate networked PCs. For multiple cells, OpenBSC and a first Osmo-USRP instance run on one PC while a second Osmo-USRP instance runs on a separate networked PC. All configurations are tested using a USRP 1 with two RFX-900 daughterboards and a USRP 2 with a WBX daughterboard.

An important issue with MS-BTS radio synchronization is clock accuracy due to oscillator drift. The clock accuracy required by GSM specification [27] is 0.05 parts per million (ppm), or 0.1 ppm for a pico class BTS. The MS uses the Frequency Correction Channel

(FCCH) to lock its internal clock to that of the BTS, and all other carriers on that BTS are derived from that same source clock. The standard USRP clock does not meet the GSM requirements, although the FCCH can still be locked onto when the MS searches for carriers despite the clock offset. However, if multiple BTSs in a network are not synchronized, the carriers of neighbor cells are not found by the MS if there is any significant clock offset; the neighbor cell carrier will not be located at the frequency expected by the MS and no search is undergone. Therefore, a shared external reference clock (locked from a function generator) is used in this work for testing handover to reduce the combined clock offset between two USRPs.

By implementing the Osmo-USRP interface and testing the resulting GSM network, a foundational program is provided for further testing and development by the open-source community. More users can operate and develop OpenBSC-related projects with the inexpensive and available USRP hardware. Other users can connect OpenBTS to a GSM core network using the A-bis-over-IP interface of OsmoBTS. The overall objective is increased functionality and network configuration options for running a fully open-source GSM network.

4. Implementation

This section provides details of the design of the Osmo-USRP software architecture, while describing the inter-layer primitives and modifications of the existing OpenBTS and OsmoBTS programs.

4.1 Osmo-USRP Architecture

The variation of system architectures of OpenBTS and OsmoBTS has been a major obstacle to progressing development in the Osmocom community. In OpenBTS, each of the GSM logical channels processes inter-layer messages in an individual thread. However, OsmoBTS features an event-driven design that handles all messages consecutively. Osmo-USRP uses inter-layer primitives already defined in OsmoBTS to process messages passed between the multi-threaded L1 and the event-driven L2 via Unix domain sockets. In the downstream direction, a thread multiplexer functions to route the message to the appropriate OpenBTS L1 logical channel specified by the L2 header. In the reverse direction, L2 header information is attached to L1 messages, which are then written to the upstream socket and processed in OsmoBTS L2.

The Osmo-USRP interface is designed be compatible with mainstream OsmoBTS, so OpenBTS is heavily adapted. The L2 and hybrid L3 is stripped from OpenBTS, leaving the remaining Transceiver, L1 processors, and logical channels. The original L1 processors still connect to the Transceiver, which organizes radio bursts and interfaces with the USRP hardware. Each logical channel, with its own L1 processor depending on channel type, feeds into the thread multiplexer. This system architecture, along with the connections to USRP and OsmoBTS, is shown in Figure 4.1.

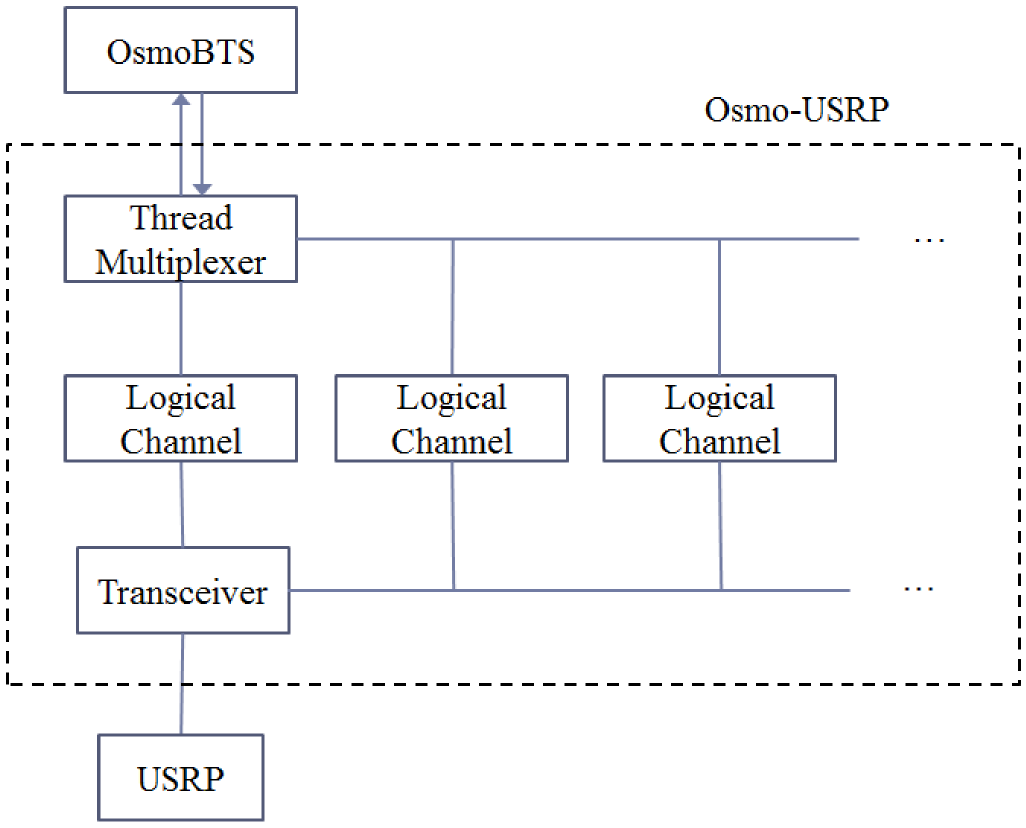

Figure 4.1: Basic system architecture and interface connections of Osmo-USRP.

4.2 OpenBTS Modifications

As previously mentioned, most of the modifications to existing code are implemented in OpenBTS. Although the existing framework is very incomplete, the OpenBTS-Osmo project is used as a foundation for this work. OpenBTS-Osmo severs L2 and up from the required L1 code and also contains an empty skeleton of logical channels and thread multiplexer that are filled in with functionality in Osmo-USRP [28]. The details of the specific changes are described in this section.

4.2.1 Thread Multiplexer

Aside from some missing L1 functionality and logical channel structuring, the thread multiplexer entity contains most of the processing added in Osmo-USRP. When the Osmo-USRP program starts, four First In First Out (FIFO) files are created in the "/dev/msgq/" system directory, which are required and opened by OsmoBTS to communicate between processes via Unix domain sockets. There are two socket pairs for sending separate L1 and System primitive messages in both directions.

Four threads are spawned at start-up, two for receiving messages on the L1 and System sockets, one for sending messages on the L1 socket, and one for sending MphTimeInd messages to L2 every TDMA frame. The thread for sending L1 type messages contains a FIFO queue for aggregating all outgoing messages from multiple logical channels. However, a separate thread for sending messages on the System socket is not required because this type of message is processed infrequently.

The MphTimeInd loop is started when the Synchronization Channel (SCH) is activated and stopped when the SCH is deactivated. While active, OsmoBTS runs a "keep alive" timer that ensures L1 is responsive; if no MphTimeInd message is received after five seconds, OsmoBTS is terminated. Additionally, OsmoBTS requires the included frame number to know when to compute uplink channel measurements that have been aggregated over a full measurement period. These uplink measurements are acquired over a SACCH multiframe (480 ms) for TCH and SDCCH logical channels, or 104 frames and 102 frames respectively [25]. The average RXLEV (received signal strength) and RXQUAL (frame error rate) values are then sent to OpenBSC in a MEASurement RESult message on the RSL of A-bis L3 [29].

Another issue is the conversion between two data structures, BitVector of OpenBTS and msgb of OsmoBTS. For incoming PhDataReq messages from L2, the msgb contains a "msgUnitParam" field with a byte buffer and corresponding size. If the logical channel is TCH, Osmo-USRP packs the 33 byte buffer into a VocoderFrame data structure and adds it to a FIFO queue of speech frames in the TCH L1 processor. If the incoming data message is for any other logical channel, the 23 byte buffer is packed into a BitVector which is added to an L2Frame data structure and sent to the appropriate L1 processor. Further details of L1 processing are found in Subsection 4.2.3.

4.2.2 Logical Channels

Logical channels are responsible for organizing the flow of messages between L2 and target L1 processors. In OpenBTS, a logical channel served as an inter-layer connection between L1 processor, Service Access Point (SAP) multiplexer, and L2 LAPDm processor. Each logical channel in Osmo-USRP, which simply contains an L1 processor that runs in its own thread, corresponds to one of the same type in OsmoBTS. The thread multiplexer holds references to all logical channels within the eight timeslots preconfigured in Osmo-USRP. The only channel combinations currently supported are C1, C5, and C7, which are previously described in Subsection 2.3.2.

There are several special cases in the organization of these logical channels. Since the FCCH processor simply transmits frequency correction bursts (FB) consisting of all zero-value bits in order to generate a pure sine wave [5], the L1 encoder runs in a self-contained loop with no logical channel. The loop can only be started and stopped via MphActivateReq and MphDeactivateReq messages, and it does not receive any L2 frames. A full-rate TCH and its

Fast Associated CCH (FACCH) are contained within a single logical channel, although there are separate transport functions for TCH speech frames and FACCH frames. OsmoBTS primitive messages intended for the FACCH are ignored because identical TCH messages are also sent and processed instead. Finally, each SDCCH or TCH logical channel contains a reference to its sibling SACCH, and vice versa; this organization ensures that a processed SACCH message is always associated with a corresponding sibling channel.

4.2.3 L1 Processors

As previously mentioned, each logical channel contains an L1 processor running in its own thread. The L1 processor is composed of at least one of two entities depending on the directionality of the specific channel type. These encoders and decoders operate according to the Forward Error Correction (FEC) channel coding rules for their specific channel type [30]. For the uplink direction, FEC decoding transforms each group of four received normal bursts (NB) into an L2 frame. For the downlink direction, a FEC encoder generates four NBs from each L2 frame. Unlike NBs grouped into fours, there are other single burst types used on certain logical channels. A FCCH encoder generates FBs, a SCH encoder generates synchronization bursts (SB), and a RACH decoder handles uplink access bursts (AB) [5]. Downlink dummy bursts, used for idle transmission when there are no bursts from L1, are taken from a filler burst table in the Transceiver.

Whereas OpenBTS directly generates SBs for the SCH and NBs for the BCCH based on preconfigured L3 information, Osmo-USRP is adapted to receive this L3 information from OsmoBTS. Standard L2 frames are sent to the appropriate BCCH logical channel and processed by the typical L1 encoder for NBs. The L2 message for the SCH has only four bytes, which are processed by a new L1 encoder designed for SBs.

In order to support the uplink channel measurements described in Section 3, a L1 decoder acquires three parameters from each radio burst received from the Transceiver. These physical channel parameters are taken from a single AB or averaged over four NBs: received signal strength indication (RSSI) which corresponds to the signal power at the BTS; timing advance (TA) which corresponds to the propagation time of the signal from MS to BTS; and frame error rate (FER) which corresponds to the quality of the signal at the BTS. FER is determined by a 40-bit parity check [30] of each burst, aggregated over a sliding window with a length of 10 frames (known as FER decay time). The channel measurements are added to the "measParam" field of the msgb for PhDataInd and PhRaInd primitive messages. As previously mentioned, the PhDataInd measurements are only processed for RXLEV (RSSI) and RXQUAL (FER) values on TCH and SDCCH channels; the PhRaInd includes the TA for further use in the IMMediate ASSignment procedure [10].

For inter-layer flow control [9] in the downlink direction, OsmoBTS also expects a PhReadyToSendInd primitive message to know when to pull a L2 message from a LAPDm queue or create a new PhDataReq message. Then the PhDataReq, or PhEmptyFrameReq when a TCH or FACCH has no L2 frame queued, is written to the downlink L1 socket. The PhEmptyFrameReq messages are ignored by the thread multiplexer because TCH and FACCH filler bursts are generated in the associated L1 encoder when there is no VocoderFrame or L2Frame in the TCH or FACCH message queues respectively. The cycle of PhReadyToSend and PhDataReq begins when the logical channel is activated and ends when the logical channel is deactivated. After each PhDataReq is processed by the L1 encoder, the next PhReadyToSend signal is triggered for that logical channel, and the following PhDataReq is received in time to be transmitted at the requested frame. For the uplink direction, PhDataInd messages are simply sent to OsmoBTS L2 as they arrive on the Um radio interface. Figure 4.2 shows a diagram of these data flow controls in both the downlink and uplink directions of the L1-L2 interface.

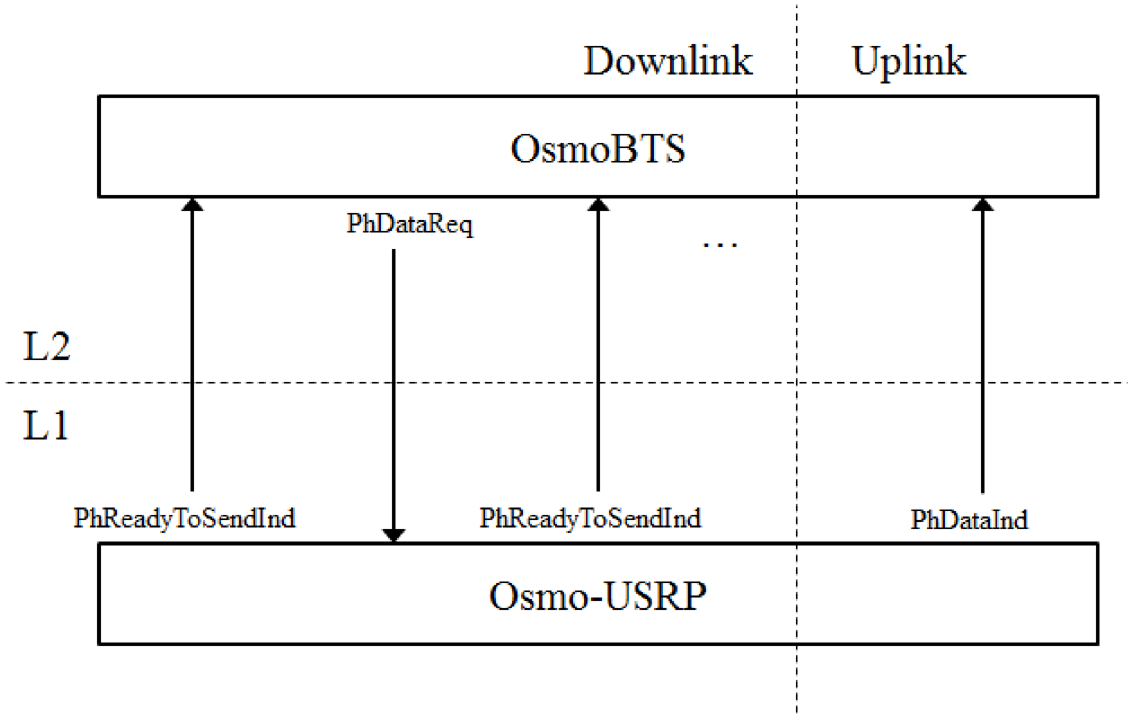

Figure 4.2: Inter-layer flow control between Osmo-USRP and OsmoBTS.

4.3 L1-L2 Primitives

There are two groups of primitives for sending messages between L1 and L2, System and L1 (respectively named "FemtoBts" and "GsmL1" in OsmoBTS). These inter-layer primitives are not publicly released but are reverse-engineered by looking at the messages sent and received by OsmoBTS. An L1-L2 primitive is one of three types similar to the L2-L3 primitive types defined in [9]: REQuest, CONFirm, and INDication. A REQ message is sent from L2 to L1 to request a service. A CONF message is returned from L1 to L2 to acknowledge that a REQ message was received. An IND message is similar to a REQ message but sent from L1 to L2 to request a service. The fourth type of primitive, RESPonse, is unnecessary and not included in OsmoBTS due to the reliable inter-layer connectivity. CONF messages in OsmoBTS typically include a status indicating how the processing ended, such as “success”, “uninitialized”, “unsupported”, or “already deactivated”.

Subsections 4.3.1-2 cover the descriptions and typical processing of System and L1 primitives. Subsection 4.3.3 is concerned with the special cases of processing some L1 primitives in the Osmo-USRP thread multiplexer.

4.3.1 System Primitives

OsmoBTS sends System primitives in order to initialize and configure the originally intended L1 hardware. Since the USRP hardware is preconfigured by the Transceiver and “main()” function when Osmo-USRP starts, Osmo-USRP performs minimal to no processing depending on the specific primitive. The used System primitives and how they are processed in Osmo-USRP are listed in Table 4.1. These System primitives could be used to control low-level USRP functions, such as activating and deactivating the RF transmitter, in future work.

Table 4.1: System primitive types used in the Osmo-USRP interface.

Primitive Received from L2	**Primitive Sent to L2**	**Processing**
SystemInfoReq	SystemInfoCnf	Return supported GSM band and arbitrary hardware versions
ActivateRfReq	ActivateRfCnf	Return success status
DeactivateRfReq	DeactivateRfCnf	Return success status
SetTraceFlagsReq	None	None
Layer1ResetReq	Layer1ResetCnf	Return success status

4.3.2 L1 Primitives

L1 primitives in OsmoBTS are used to configure the radio parameters, timeslot combinations, and logical channels of L1, as well as transport data frames in both directions. Several messages include L1 and L2 IDs, which are used as unique handles to refer to a specific L1 (Osmo-USRP thread multiplexer) or L2 (Osmo-BTS logical channel) entity. The L1 ID is specified in the MphInitCnf message and stored in OsmoBTS for subsequent messages. The used L1 primitives and how they are processed in Osmo-USRP are described in Table 4.2. Since the "main()" function preconfigures the radio parameters and timeslot combinations when Osmo-USRP starts, MphInitReq and MphConnectReq processing ignores the incoming configuration values defined in OpenBSC. These L1 primitives could be used to initialize radio and timeslot configurations in future work, while removing hard-coded and redundant values from Osmo-USRP configuration.

Table 4.2: L1 primitive types used in the Osmo-USRP interface.

Primitive Received from L2	**Primitive Sent to L2**	**Processing**
MphInitReq	MphInitCnf	Return L1 ID and success status
MphConnectReq	MphConnectCnf	Check L1 ID, return success status
MphConfigReq	MphConfigCnf	Check L1 ID, store payload type for TCH, return appropriate status
MphActivateReq	MphActivateCnf	Check L1 ID, store L2 ID in specified logical channel and start its L1 processor, signal to return PhReadyToSendInd, store payload type for TCH, return appropriate status
MphDeactivateReq	MphDeactivateCnf	Check L1 ID, clear L2 ID in specified logical channel and stop its L1 processor, return appropriate status
PhDataReq	None	Check L1 ID, send data buffer to L1 processor of specified logical channel
PhEmptyFrameReq	None	Check L1 ID, check if specified logical channel is TCH or FACCH
None	PhRaInd	Send channel measurements, RA data, and L2 ID of logical channel
None	PhDataInd	Send channel measurements, data, and L2 ID of logical channel
None	PhReadyToSendInd	Send L1 ID, requested GSM time, and L2 ID of logical channel
None	MphTimeInd	Send current GSM time

4.3.3 Special Cases for L1 Primitives

There are several L1 primitive messages that require atypical processing depending on the associated logical channel.

The MphActivateReq processor contains special handling for five certain logical channels. Since Osmo-USRP has no logical channel for FCCH, the L1 processor is started directly. Also, since a TCH and FACCH share a logical channel in Osmo-USRP, an L2 REQuest to activate a FACCH is ignored and replied with a “success” status. If the logical channel type is a SACCH, the L2 ID is checked with that of the SACCH's sibling logical channel. Activating the SCH logical channel starts the loop for sending MphTimeInd messages to L2. Finally, if the RACH is activated, no signal is triggered for starting the PhReadyToSendInd cycle because the RACH logical channel functions only in the uplink direction.

MphDeactivateReq has special cases similar to MphActivateReq: for FCCH, the L1 processor is stopped directly; for FACCH, deactivation is ignored and a "success" status is replied; and for SCH, the loop for sending MphTimeInd messages is stopped.

Both PhDataReq and PhDataInd processors handle special frame formats for the TCH and SACCH data buffers; these unique formats allow OsmoBTS to accommodate the originally intended L1 hardware. For a TCH frame, there is an extra byte inserted before the standard byte buffer that indicates the payload type of the logical channel; in the PhDataReq case, this payload type is checked against the configured payload type of the receiving logical channel. For a SACCH frame, there are two extra bytes, inserted before the standard byte buffer, that are reserved for MS power and timing measurements.

4.4 OsmoBTS Modifications

Although Osmo-USRP contains most of the design implementations of this work, OsmoBTS is slightly modified from its mainstream branch. The correct number of bytes is read from the upstream sockets depending on the primitive type, since L1 and System messages are different lengths. The System Information (SI) messages [5] generated for BCCH L2 frames repeat in a 1-2-3-4-3-2-3-4 pattern, as originally done in the BCCH L1 encoder of OpenBTS. A bug fix with merging BTS attributes on the OML of A-bis L3 is reverted to bypass a segmentation fault error; this reversion does not have an effect on any tested OsmoBTS operation. These minor changes are necessary for running Osmo-USRP as a L1 hardware entity [31].

5. Results

The configurations and results of testing the Osmo-USRP program for single- and multiple-cell networks are covered in this section. Details of the intra-BSC handover procedure are also discussed.

Figure 5.1 shows the laboratory test setup that is used for the results of this work. The USRP 1 and USRP 2 hardware, two MS handsets, and a laptop running four terminals with separate processes: Osmo-USRP, OsmoBTS, OpenBSC, and the OpenBSC VTY interface.

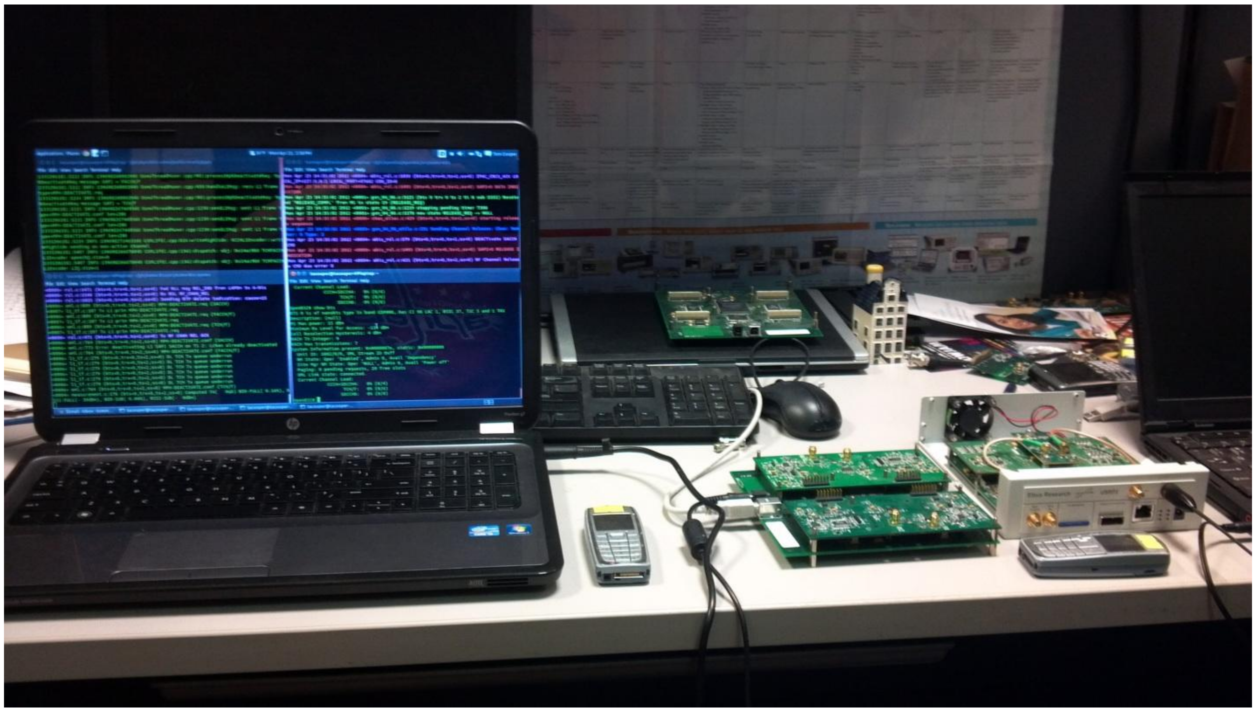

Figure 5.1: Laboratory test setup for Osmo-USRP operation.

5.1 Single Cell

In order to test the operation of the Osmo-USRP program in a GSM network, a setup of open-source software and hardware similar to Figure 3.1 is configured. Single cell operation is tested using the Osmo-NITB application of OpenBSC, a USRP 1 with two RFX-900 daughterboards, and several Nokia 3120 mobile handsets. Tests are conducted in the EGSM-900 band on ARFCN 61. Typical GSM events are completed successfully, including subscriber authentication and encryption, location updating, IMSI attach and detach, SMS storage and delivery, voice calling with the full-rate audio codec, and uplink and downlink measurement reporting. SMS messaging and voice calls are conducted using the appropriate UI functions on the MS. Detailed instructions for installation, configuration, and operation are found in the Appendices.

5.2 Multiple Cells

For multi-cellular network operation, a second computer with the Osmo-USRP program and a USRP 2 with WBX daughterboard are used in addition to the single cell configuration. The second Osmo-USRP instance is connected to Osmo-NITB on the first computer. The two USRP hardware setups are located approximately 5 m apart from each other. Tests are conducted in the EGSM-900 band on ARFCNs 51 and 61. Again, typical GSM events are completed successfully, including SMS delivery, paging, IMSI attach and detach, and voice calling across the two cells.

5.3 Intra-BSC Handover

Additionally, intra-BSC (or inter-cell) handover [26] is tested in the multi-cellular configuration. In terms of physical channel establishment, the non-synchronized cell case [10] is considered for an active TCH. Figure 5.2 describes the intra-BSC handover process, based on [32] and [33], where the MS is moving from BTS 1 to BTS 2 during a voice call. The BSC receives MEASurement RESult messages that contain downlink and uplink measurements on the current and neighboring cells, and decides to relocate the MS from BTS 1 to a new cell. The new TCH, which is configured to process L1 ABs, is allocated on the neighboring BTS 2 and the Handover Command is sent to the MS on the FACCH of BTS 1. The MS tunes to the new TCH, but it does not know the TA value yet and therefore is not time synchronized, so Handover Access messages are sent as ABs. When an AB is detected on the new TCH, the BSC reroutes the voice path through BTS 2. A PHYsical INFOrmation message is sent to the MS containing TA and frequency corrections, after which the MS can send TCH speech frames as NBs. A Set Asynchronous Balanced Mode (SABM) and Unnumbered Acknowledge (UA) handshake establishes the LAPDm signaling connection. The MS responds with a Handover Complete message and the BSC finally releases the old TCH on BTS 1.

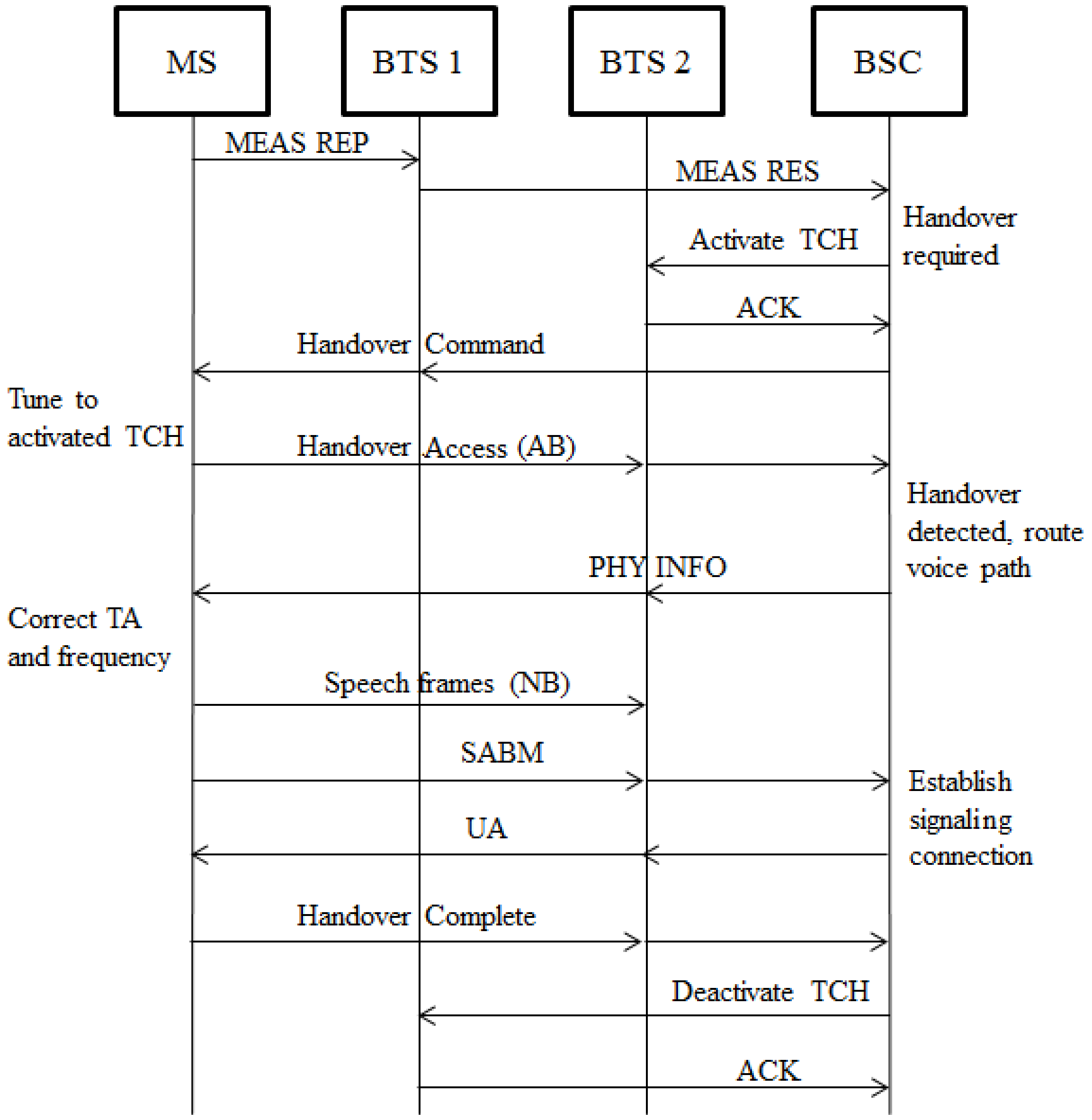

Figure 5.2: Intra-BSC handover procedure.

This procedure is unsuccessful due to lack of handover processing in L1 and L2 of Osmo-USRP and Osmo-BTS. Mainly, the TCH in Osmo-USRP is not properly configured to handle Handover Access ABs from the MS. Section 6.2 provides more details of the proposed handover implementation in L1 and L2.

6. Discussion

This section provides a discussion of the results and limitations of the Osmo-USRP program, as well as proposed implementations of these limitations for guidance in future work.

6.1 Limitations

As described in the implementation and results sections, most of the functionality of a standard BTS is present in Osmo-USRP and OsmoBTS. However, the OpenBTS L1 only supports full-rate speech, and GPRS data is not supported. Multiple ARFCNs, intra-BTS handover, and intra-BSC handover are also not implemented. Osmo-USRP also uses only two of three available CCCHs in the C5 timeslot configuration, one for the AGCH and one for the PCH. Many of the network and configuration values are hard-coded in Osmo-USRP and not dynamically controlled by OpenBSC.

Occasionally, GSM events experience unexpected failure during testing. The MS does not access the network, the BTS does not release SDCCH or TCH resources after their operation has ended, or voice calls are not routed properly through the BTS. The reliability of the tested MSs is one reason for unexpected behavior; sometimes only certain handsets can access the test network, and only a subset of those can make voice calls or send SMS messages. There are also certain to be unknown bugs in the code at this early stage of development and testing. As the Osmocom and OpenBTS communities use Osmo-USRP more extensively, these bugs and hard-coded configuration values will be fixed; development is an ongoing process that requires testing of all possible configurations.

6.2 Future Work

The primary concerns for future work on Osmo-USRP are increasing the operational reliability and implementing missing BTS functionality. For the former, more network configurations should be tested, including a network of three or more cells. Testing and development by the open-source community would be encouraged if the code is eventually integrated into the official OpenBTS-Osmo project as expected; however, it remains to be seen if OpenBTS-Osmo will be developed separately or jointly with Osmo-USRP.

The main limitation needing implementation is the intra-BSC handover procedure in L1 and L2. In OsmoBTS, the RSL of L3 receives a CHANnel ACTIVation REQuest [29], which needs to be checked for an Activation Type value related to inter-cell channel change. If the activation is handover related, OsmoBTS sends a request to Osmo-USRP to activate the RACH L1 processor on the new TCH logical channel; this activation is performed using a new L1-L2 primitive or modifying the current MphActivateReq primitive type. Then Osmo-USRP receives an AB on this TCH and sends the PhRaInd message to OsmoBTS, where it is processed as a Handover Access message. Finally, OsmoBTS deactivates the RACH L1 processor and activates the normal TCH/FACCH and SACCH logical channels required for a call in Osmo-USRP. At this point, the standard L2 establishment continues as it is currently implemented. As indicated, Osmo-USRP must also be restructured to support assigning a certain type of L1 processor to a specific logical channel; currently, the type of L1 processor that a logical channel contains is hard-coded. Other modifications in the ARFCN manager are required to dynamically overwrite the demultiplexing table when installing a new RACH L1 decoder in a TCH timeslot. As of now, the demultiplexing table is statically initialized and decoder types are limited to certain frames, since decoding ABs on a TCH timeslot during handover is an unusual circumstance that is not accounted for in the original L1 of OpenBTS.

Additionally, the management of CCCHs can be improved by using a pool system similar to that of OpenBTS. When an AGCH or PCH is needed, an available CCCH is chosen based on minimum load, and the message is sent to that CCCH. The current load of a CCCH is determined by the number of messages waiting in the CCCH's transmission queue. The number of CCCHs used for each pool of AGCHs and PCHs can be configured at initialization.

The issues of hard-coded parameters during initialization and using three separate configuration files can be remedied. Many values are currently repeated in the Osmo-USRP and OpenBSC configuration files; the OsmoBTS configuration file contains minimal values for identifying the BTS on a network. When the values are received in Osmo-USRP from OpenBSC, they are ignored because the radio parameters and logical channel configurations have already been set at startup. Osmo-USRP can be restructured to only start the thread multiplexer and wait on configuration messages from OsmoBTS to initialize timeslot combinations, radio parameters such as the ARFCN, and network values such as the BSIC. Ideally, only the OpenBSC and minimal OsmoBTS configuration files are needed since OsmoBTS and Osmo-USRP run on the same system.

7. Conclusion

Osmo-USRP provides a method of connecting an open-source software BTS L1 to an open-source GSM network that previously included only L2-L3 of a software BTS. Standard GSM events function at a reasonably reliable level for single cell and multi-cellular network configurations. Much of this work has involved understanding the GSM technical specifications and procedures for these events, as well as how they are processed on the L1-L2 interface of the BTS. The development of an interface to an existing program has provided insight to the methods of controlling data flow between layers. The management of messages intended for specific logical channels is based on predefined primitives that inform the thread multiplexer of the correct processing function. Message FIFO queues are also used for asynchronous inter-process communication between Osmo-USRP and OsmoBTS. Ready-to-send indication messages notify the upper layer that the lower layer is prepared to process a new data request. By probing the OsmoBTS L2-L1 request messages and the processing function of the expected reply messages, the correct responses are built in Osmo-USRP L1. This method of analyzing and constructing the L1-L2 primitive messages has provided a deeper understanding of reverse-engineering an inter-layer protocol.

Ideally, the Osmo-USRP interface will be tested on a larger scale network with many cells using the Virginia Tech Cognitive Radio Network Testbed (VTCORNET) [34]. The VTCORNET system provides a highly configurable method of operating multiple BTS radios in different locations, which is helpful for testing GSM network-related events such as intra-BSC handover.

In the current push towards a fully open, low-cost mobile network, every component offers a significant contribution. Many of the developers in the Osmocom community are in need of cheaper, readily available hardware. Commercial BTS vendors do not sell single units to hobbyists and researchers; however, the USRP has emerged as a prominent choice for building cheap SDRs. Osmo-USRP allows Osmocom developers to use the USRP, much like the many OpenBTS users. OpenBTS users can also now deploy larger multi-cellular networks by using Osmo-USRP with OpenBSC.

Although there is still future work to be done for the noted limitations, this project serves as a foundation for developers to expand on through their own contributions. As mentioned in Section 6.2, inter-cell handover is the next logical functionality to implement. Applications related to cognitive radio networks are enabled by the fully open-source BTS stack with handover support. For instance, dynamic spectrum access (DSA) techniques can be researched and implemented using frequency hopping in coordination with the BSC; this functionality allows a multi-cellular GSM network to operate in unlicensed "white spaces" spectrum. Additionally, research areas such as indoor position location and anticipatory handover for multi-access networks [35] can be investigated in the context of GSM networks. These example applications show how one can use the BTS stack enabled by Osmo-USRP in order to customize an open-source GSM network for future research and development.

Bibliography

[1] D. Burgess and H. Samra. (2008, Oct.) The OpenBTS Project. [Online]. Available: http://www.ahzf.de/itstuff/papers/OpenBTSProject.pdf

[2] GSMA. (2010, Jul.) GSMA Announces that Global Mobile Connections Surpass 5 Billion. [Online]. Available: http://www.gsma.com/articles/gsma-announces-that-global-mobile-connections-surpass-5-billion/17632

[3] Range Networks. [Online]. Available: http://www.rangenetworks.com

[4] 3GPP, "Mobile Radio Interface Signalling Layer 3; General Aspects," TS 04.07, v7.3.0, Dec. 1999.

[5] 3GPP, "Multiplexing and Multiple Access on the Radio Path," TS 05.02, v8.11.0, Jun. 2003.

[6] 3GPP, "Mobile Station - Base Station System (MS - BSS) Interface; Channel Structures and Access Capabilities," TS 04.03, v8.0.2, May 2002.

[7] J. Eberspächer, H.-J. Vögel, C. Bettstetter, and C. Hartmann, *GSM Switching, Services, and Protocols,* 3rd ed. West Sussex, United Kingdom: John Wiley and Sons, 2009. ISBN: 0-4700-3070-4.

[8] GSM Logical Channels. [Online]. Available: http://www.gsmfordummies.com/tdma/logical.shtml

[9] 3GPP, "Mobile Station - Base Station System (MS - BSS) Interface; Data Link (DL) Layer Specification," TS 04.06, v8.4.0, Dec. 2008.

[10] 3GPP, "Mobile Radio Interface Layer 3 Specification," TS 04.08, v7.21.0, Dec. 2003.

[11] 3GPP, "Point to Point (PP) Short Message Service (SMS) Support on Mobile Radio Interface," TS 04.11, v7.1.0, Sep. 2000.

[12] 3GPP, "Location Services (LCS); Mobile Radio Interface Layer 3 Location Services (LCS) Specification," TS 04.71, v8.4.0, Jul. 2002.

[13] Osmocom OpenBSC. [Online]. Available: http://openbsc.osmocom.org

[14] OsmoBTS. [Online]. Available: http://openbsc.osmocom.org/trac/wiki/OsmoBTS

[15] OpenBTS-Osmo. [Online]. Available: http://cgit.osmocom.org/cgit/openbts-osmo

[16] OsmocomBB. [Online]. Available: http://bb.osmocom.org

[17] Libosmocore. [Online]. Available: http://bb.osmocom.org/trac/wiki/libosmocore

[18] Libosmo-Abis. [Online]. Available: http://openbsc.osmocom.org/trac/wiki/libosmo-abis

[19] OsmoSGSN. [Online]. Available: http://openbsc.osmocom.org/trac/wiki/osmo-sgsn

[20] OpenGGSN. [Online]. Available: http://openbsc.osmocom.org/trac/wiki/OpenGGSN

[21] Ettus Research. [Online]. Available: http://www.ettus.com

[22] GNU Radio. [Online]. Available: http://gnuradio.org

[23] OpenBTS. [Online]. Available: http://gnuradio.org/redmine/projects/gnuradio/wiki/OpenBTSBackground

[24] 3GPP, "Full Rate Speech; Transcoding," TS 06.10, v8.2.0, Jun. 2001.

[25] 3GPP, "Radio Subsystem Link Control," TS 05.08, v8.23.0, Nov. 2005.

[26] 3GPP, "Handover Procedures," TS 03.09, v7.0.0, Aug. 1999.

[27] 3GPP, "Radio Subsystem Synchronization," TS 45.010, v10.1.0, Mar. 2011.

[28] Osmo-USRP. [Online]. Available: http://github.com/tacooper/Osmo-USRP

[29] 3GPP, "Base Station Controller - Base Transceiver Station (BSC-BTS) Interface; Layer 3 Specification," TS 08.58, v8.6.0, Nov. 2000.

[30] 3GPP, "Channel Coding," TS 05.03, v8.9.0, Jan. 2005.

[31] OsmoBTS Modifications. [Online]. Available: http://github.com/tacooper/osmo-bts

[32] GSM Overview 11. [Online]. Available: http://www.hexazona.com/nexwave/docs/training/GSM%20Overview%2011.pdf

[33] GSM Handover Call Flow (Intra MSC). [Online]. Available: http://www.eventhelix.com/realtimemantra/telecom/GSM_Handover_Call_Flow.pdf

[34] Virginia Tech Cognitive Radio Network Testbed (VTCORNET). [Online]. Available: http://cornet.wireless.vt.edu/trac/wiki/CORNET

[35] V. A. de Sousa Jr., R. A. de O. Neto, F. de S. Chaves, L. S. Cardoso, and F. R. P. Cavalcanti, "Access Selection with Anticipatory Vertical Handover for Multi-Access Networks," *IEEE 17th International Symposium on Personal, Indoor and Mobile Radio Communications*, pp. 1-5, Sep. 2006.

Appendix A: Software Guide

A.1 Installation

The programs needed for this work should be installed in the order given in Subsections A.1.1-5. There are a few options for the "configure" step of Osmo-USRP installation. GNU Radio must be installed when using a USRP 1; for other USRP devices, UHD must be installed. Flags are used to specify certain setups: "--with-usrp1" for USRP 1, "--with-extref" for an external reference clock, and "--with-resamp" for 64 MHz clock rate. When resampling 64 MHz to 52 MHz, the value of masterClockRate in Transceiver52M/USRPDevice.cpp should be 64.0e6 (which is the default); otherwise use 52.0e6.

It is strongly recommended to use an external reference clock instead of the USRP stock clock in order to meet GSM specifications and ensure stable BTS operation.

A.1.1 Libosmocore

```
git clone git://git.osmocom.org/libosmocore.git
cd libosmocore
autoreconf -i
./configure
make
sudo make install
cd ..
```

A.1.2 Libosmo-Abis

```
git clone git://git.osmocom.org/libosmo-abis.git

cd libosmo-abis

autoreconf -i

./configure

make

sudo make install

cd ..
```

A.1.3 OpenBSC

```
git clone git://git.osmocom.org/openbsc.git

cd openbsc/openbsc

autoreconf -i

./configure

make

sudo make install

cd ../..
```

A.1.4 OsmoBTS

```
git clone git://github.com/tacooper/osmo-bts.git

cd osmo-bts

autoreconf -i

./configure --enable-sysmocom-bts
```

```
make

sudo make install

cd ..
```

A.1.5 Osmo-USRP

```
git clone git://github.com/tacooper/Osmo-USRP.git

cd Osmo-USRP/public-trunk

autoreconf -i

./configure --with-usrp1 --with-resamp

make

sudo make install

cd ../..
```

A.2 Operation

Operation must be conducted in four separate Terminal windows. These Terminals are used for running the programs in Subsections A.2.1-4. Note that “sudo” is required for running OsmoBTS and Osmo-USRP because of the inter-process FIFO files created in /dev/msgq.

Also, many of the values must be repeated across configuration files because they are not dynamically configured in Osmo-USRP by OpenBSC.

A.2.1 OpenBSC

```
cd openbsc/openbsc/src/osmo-nitb
```

<u>For operating single BTS:</u>

Place the file and edit the values in openbsc.cfg (Section A.3).

```
./osmo-nitb --debug=DRLL:DCC:DMM:DRR:DRSL:DNM:DLMUX:DHO -T -e 0
     -c openbsc.cfg
```

<u>For operating multiple BTSs:</u>

Place the file and edit the values in multi-bts.cfg (Section A.3).

```
./osmo-nitb --debug=DRLL:DCC:DMM:DRR:DRSL:DNM:DLMUX:DHO -T -e 0
     -c multi-bts.cfg
```

<u>For testing handover with multiple BTSs:</u>

Place the file and edit the values in multi-bts.cfg (Section A.3); be sure to enter a value of 1 next to the “handover” parameter. Also include the -P flag to enable RTP proxy mode needed for handover.

```
./osmo-nitb --debug=DRLL:DCC:DMM:DRR:DRSL:DNM:DLMUX:DHO -T -e 0
     -c multi-bts.cfg -P
```

A.2.2 OsmoBTS

```
cd osmo-bts/src/osmo-bts-sysmo
```

Edit the values in the included file, osmo-bts.cfg.

```
sudo ./sysmobts
```

OsmoBTS will not function until the four FIFO files are created by Osmo-USRP. Often, this step requires running Osmo-USRP first, and then repeatedly running OsmoBTS until the main program starts (typically 4-5 iterations).

A.2.3 Osmo-USRP

```
cd Osmo-USRP/public-trunk/apps
```

Edit the values in the included file, TrueBTS.config.

```
sudo ./TrueBTS
```

As mentioned previously, Osmo-USRP is often required to run while OsmoBTS is run repeatedly until the four FIFO files are opened.

A.2.4 Osmo-NITB VTY

The VTY interface allows the user to configure BSC options and inspect the GSM network.

```
telnet localhost 4242
```

Note that the BSC must have full-rate audio enabled, as it is the only codec compatible with Osmo-USRP.

```
enable

configure terminal

mncc-int

default-codec tch-f fr
```

This command shows the current configuration of the network, which is used here to check that the codec is indeed full-rate (FR).

```
show running-config
exit
```

Additionally, the HLR database must be properly configured to allow subscribers to access the network. The IMSI of the subscriber must be known; if unknown, it will be shown in the debug logging output of Osmo-NITB when the MS is rejected by the network. The following commands are then used to assign a subscriber name for ease of reference, assign a telephone extension number for calling, authorize the subscriber, and display the subscriber's information, respectively. The subscriber can also be referenced by the ID number displayed with the show command.

```
subscriber imsi 012340123456789 name Cooper
subscriber imsi 012340123456789 extension 5555
subscriber imsi 012340123456789 authorized 1
show subscriber imsi 012340123456789
```

This command is helpful for viewing the statuses of all connected BTSs. The channel load figures are of particular interest, as TCH and SDCCH slots are known to not be released after errors.

```
show bts
```

A.3 Configuration Files

These two OpenBSC configuration files are provided here since they are not part of the Osmo-USRP and OsmoBTS codebases. The specific parameters are compatible with the configuration files included in Osmo-USRP and OsmoBTS.

For operating single BTS:

```
!
! OpenBSC configuration saved from vty
!   !
password foo
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
network
 network country code 001
 mobile network code 02
 short name Osmo-USRP
 long name Osmo-USRP
 auth policy closed
 location updating reject cause 13
 encryption a5 0
 neci 1
 rrlp mode none
 mm info 1
 handover 0
 handover window rxlev averaging 10
 handover window rxqual averaging 1
 handover window rxlev neighbor averaging 10
 handover power budget interval 6
 handover power budget hysteresis 3
 handover maximum distance 9999
 timer t3101 10
 timer t3103 0
 timer t3105 0
 timer t3107 0
 timer t3109 0
 timer t3111 0
 timer t3113 60
 timer t3115 0
 timer t3117 0
 timer t3119 0
 timer t3141 0
 bts 0
  type nanobts
  band GSM900
  cell_identity 98
  location_area_code 1
  training_sequence_code 5
  base_station_id_code 37
  ms max power 15
  cell reselection hysteresis 4
  rxlev access min 0
  channel allocator ascending
  rach tx integer 9
  rach max transmission 7
  ip.access unit_id 1802 0
  oml ip.access stream_id 255 line 0
  gprs mode none
  trx 0
   rf_locked 0
   arfcn 61
   nominal power 23
   max_power_red 20
   rsl e1 tei 0
    timeslot 0
     phys_chan_config CCCH+SDCCH4
    timeslot 1
     phys_chan_config SDCCH8
    timeslot 2
     phys_chan_config TCH/F
    timeslot 3
     phys_chan_config TCH/F
    timeslot 4
     phys_chan_config TCH/F
    timeslot 5
     phys_chan_config TCH/F
    timeslot 6
     phys_chan_config TCH/F
    timeslot 7
     phys_chan_config TCH/F
```

For operating multiple BTSs:

```
!
! OpenBSC configuration saved from vty
!   !
password foo
!
line vty
 no login
!
e1_input
 e1_line 0 driver ipa
network
 network country code 001
 mobile network code 02
 short name Osmo-USRP
 long name Osmo-USRP
 auth policy closed
 location updating reject cause 13
 encryption a5 0
 neci 1
 rrlp mode none
 mm info 1
 handover 1
 handover window rxlev averaging 10
 handover window rxqual averaging 1
 handover window rxlev neighbor averaging 10
 handover power budget interval 6
 handover power budget hysteresis 3
 handover maximum distance 9999
 timer t3101 10
 timer t3103 0
 timer t3105 0
 timer t3107 0
 timer t3109 0
 timer t3111 0
 timer t3113 60
 timer t3115 0
 timer t3117 0
 timer t3119 0
 timer t3141 0
 bts 0
  type nanobts
  band GSM900
  cell_identity 99
  location_area_code 1
  training_sequence_code 5
  base_station_id_code 37
  ms max power 15
  cell reselection hysteresis 4
  rxlev access min 0
  channel allocator ascending
  rach tx integer 9
  rach max transmission 7
  ip.access unit_id 1801 0
  oml ip.access stream_id 255 line 0
  gprs mode none
  trx 0
   rf_locked 0
   arfcn 51
   nominal power 23
   max_power_red 20
   rsl e1 tei 0
    timeslot 0
     phys_chan_config CCCH+SDCCH4
    timeslot 1
     phys_chan_config SDCCH8
    timeslot 2
     phys_chan_config TCH/F
    timeslot 3
     phys_chan_config TCH/F
    timeslot 4
     phys_chan_config TCH/F
    timeslot 5
     phys_chan_config TCH/F
    timeslot 6
     phys_chan_config TCH/F
    timeslot 7
     phys_chan_config TCH/F
 bts 1
  type nanobts
  band GSM900
  cell_identity 98
  location_area_code 1
  training_sequence_code 5
  base_station_id_code 37
  ms max power 15
  cell reselection hysteresis 4
  rxlev access min 0
  channel allocator ascending
  rach tx integer 9
  rach max transmission 7
  ip.access unit_id 1802 0
  oml ip.access stream_id 255 line 0
```

```
gprs mode none
trx 0
 rf_locked 0
 arfcn 61
 nominal power 23
 max_power_red 20
 rsl e1 tei 0
  timeslot 0
   phys_chan_config CCCH+SDCCH4
  timeslot 1
   phys_chan_config SDCCH8
  timeslot 2
   phys_chan_config TCH/F
  timeslot 3
   phys_chan_config TCH/F
  timeslot 4
   phys_chan_config TCH/F
  timeslot 5
   phys_chan_config TCH/F
  timeslot 6
   phys_chan_config TCH/F
  timeslot 7
   phys_chan_config TCH/F
```