Free / Open Source Software for GSM

Harald Welte

gnumonks.org
OpenBSC
airprobe.org
osmocom.org
hmw-consulting.de

December 2010, Taiwan

Part I - Open Source GSM Tools

1. OpenBSC
   - OpenBSC Network In The Box
   - OpenBSC BSC-only mode
   - OpenBSC GPRS support
   - OpenBTS
2. OsmocomBB Project
   - OsmocomBB Introduction
   - OsmocomBB Software
   - OsmocomBB Hardware Support
   - OsmocomBB Project Status
3. wireshark Protocol Analyzer
4. Osmocom SIMtrace
   - Debugging SIM drivers and STK apps
   - Osmocom SIMtrace Introduction
   - Osmocom SIMtrace Hardware

OpenBSC software

OpenBSC is a Open Source implementation of (not only) the BSC features of a GSM network.

- Support A-bis interface over E1 and IP
- Support for BTS vendor/model is modular, currently Siemens BS-11 and ip.access nanoBTS
- Multiple BTS models/vendorrs can be mixed!
- Can work as a *pure BSC* or as a full *network in a box*
- Supports mobility management, authentication, intra-BSC hand-over, SMS, voice calls (FR/EFR/AMR)
- GPRS + EDGE support if combined with OsmoSGSN and OpenGGSN

OpenBSC

- Supports Siemens BS-11 BTS (E1) and ip.access nanoBTS (IP based)
- Has classic 2G signalling, voice and SMS support
- Implements various GSM protocols like
  - A-bis RSL (TS 08.58) and OML (TS 12.21)
  - TS 04.08 Radio Resource, Mobility Management, Call Control
  - TS 04.11 Short Message Service
- Telnet console with Cisco-style interface

OpenBSC software architecture

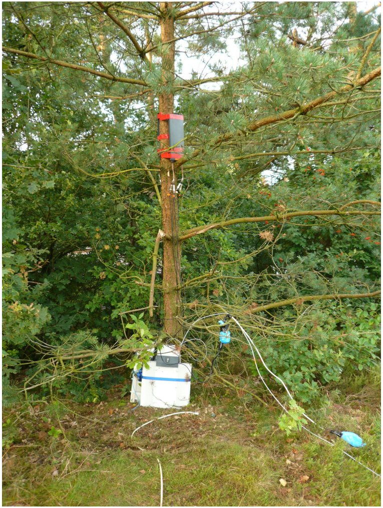
- Implemented in pure C, similarities to Linux kernel
  - Linked List handling, Timer API, coding style
- Single-threaded event-loop / state machine design
- Telnet based command line interface *Cisco-style*
- Input driver abstraction (mISDN, Abis-over-IP)

OpenBSC: GSM network protocols

The A-bis interface

Layer 1	Typically E1 line, TS 08.54
Layer 2	A variant of ISDN LAPD with fixed TEI's, TS 08.56
Layer 3	OML (Organization and Maintenance Layer, TS 12.21)
Layer 3	RSL (Radio Signalling Link, TS 08.58)
Layer 4+	transparent messages that are sent to the MS via Um

OpenBSC: Field Test at HAR2009



OpenBSC in NITB mode

Network In a Box Mode

The `bsc_hack` program

- implements the A-bis interface towards any number of BTS
- provides most typical features of a GSM network in one software
- no need for MSC, AuC, HLR, VLR, EIR, ...
    - HLR/VLR as SQLite3 table
    - Authentication + Ciphering support
    - GSM voice calls, MO/MT SMS
    - Hand-over between all BTS
    - Multiple Location Areas within one BSC

OpenBSC NITB features

OpenBSC NITB features

- Run a small GSM network with 1-n BTS and OpenBSC
- No need for MSC/HLR/AUC/...
- No need for your own SIM cards (unless crypto/auth rqd)
- Establish signalling and voice channels
- Make incoming and outgoing voice calls between phones
- Send/receive SMS between phones
- Connect to ISDN PBX or public ISDN via Linux Call Router

OpenBSC in NITB mode

Network In a Box Mode

The `bsc_hack` program

- does not implement any other GSM interfaces apart from A-bis
- no SS7 / TCAP / MAP based protocols
- no integration (roaming) with existing traditional GSM networks
- wired telephony interfacing with ISDN PBX `lcr` (Linux Call Router)
- Has been tested with up to 800 subscribers on 5 BTS
- Intended for R&D use or private PBX systems

OpenBSC LCR integration

Interfacing with wired telephony

OpenBSC (NITB mode) can be linked into Linux Call Router (`lcr`)

- OpenBSC is compiled as libbsc.a
- libbsc.a includes full OpenBSC NITB mod code
- linking the library into `lcr` results in GSM *line interfaces* to become available inside `lcr`
- OpenBSC no longer takes care of call control, but simply hands everything off to `lcr`
- Dialling plan, etc. is now configure in `lcr` like for any other wired phones

OpenBSC in BSC-only mode

The `osmo-bsc` program

- behaves like a classic GSM BSC
- uses SCCP-Lite (ip.access multipex) to any SoftMSC like ADC
- used in production/commercial deployments (75 BSCs)
- mainly intended to replace proprietary BSC in traditional GSM networks

OpenBSC

Demonstration

GPRS and OpenBSC

- The BSC doesn’t really do anything related to GPRS
- GPRS implemented in separate SGSN and GGSN nodes
- GPRS uses its own Gb interface to RAN, independent of A-bis
- OpenBSC can configure the nanoBTS for GPRS+EDGE support via OML
- Actual SGSN and GGSN implemented as OsmoSGSN and OpenGGSN programs

OsmoSGSN

The Osmocom SGSN program implements

- basic/minimal SGSN functionality
- the Gb interface (NS/BSSGP/LLC/SNDCP)
- mobility management, session management

It's a work in progress, many missing features

- no HLR integration yet
- no paging coordination with MSC/BSC
- no encryption support yet

OpenGGSN

- GPL licensed Linux program implementing GGSN node
- Implements GTP-U protocol between SGSN and GGSN
- User-configurable range/pool of IPv4 addresses for MS
- Uses `tun` device for terminating IP tunnel from MS
- provides GTP implementation as libgtp
- Experimental patches for IPv6 support

What is OpenBTS?

- is *NOT* a BTS in the typical GSM sense
- is better described as a GSM-Um to SIP gateway
- implements the GSM Um (air interface) as SDR
- uses the USRP hardware as RF interface
- does not implement any of BSC, MSC, HLR, etc.
- bridges the GSM Layer3 protocol onto SIP
- uses SIP switch (like Asterisk) for switching calls + SMS
- is developed as C++ program and runs on Linux + MacOS

What is OpenBTS?

- Open implementation of Um L1 & L2, an all-software BTS.
- L1/L2 design based on an object-oriented dataflow approach.
- Includes L3 RR functions normally found in BSC.
- Uses SIP PBX for MM and CC functions, eliminating the conventional GSM network. L3 is like an ISDN/SIP gateway.
- Intended for use in low-cost and rapidly-deployed communications networks, but can be used for experiments (including by Chris Pagent at Def Con).

OpenBTS Hardware

OpenBTS supports the following SDR hardware

- Ettus USRP(1) with two RFX 900 or RFX 1800 daughter boards
  - Modification for external clock input recommended
  - External 52 MHz precision clock recommended
- Kestrel Signal Processing / Range Networks custom radio
- Close Haul Communications / GAPfiller (work in progress)
- Ported to other radios by other clients.

OpenBTS History + Tests

- Started work in Aug 2007, first call in Jan 2008, first SMS in Dec 2008.
- First public release in September 2008, assigned to FSF in Oct 2008.
- Ran 3-sector 3-TRX system with 10,000-20,000 handsets at Sept 2009 Burning Man event in Nevada.
- Ran 2-sector 5-TRX system with 40,000 handsets at Sept 2010 Burning Man event in Nevada.
- Release 2.5 is about 13k lines of C++.
- Part of GNU Radio project, distributed under AGPLv3.
- Range Networks launched in Sept 2010 to produce commercial products and distributions.

Burning Man 2010 Tower Base

Requirements for GSM security analysis

What do we need for protocol-level security analysis?

- A GSM MS-side baseband chipset under our control
- A Layer1 that we can use to generate arbitrary L1 frames
- A Layer2 protocol implementation that we can use + modify
- A Layer3 protocol implementation that we can use + modify

None of those components existed, so we need to create them!

A GSM baseband under our control

The two different DIY approaches

- Build something using generic components (DSP, CPU, ADC, FPGA)
  - No reverse engineering required
  - A lot of work in hardware design + debugging
  - Hardware will be low-quantity and thus expensive
- Build something using existing baseband chipset
  - Reverse engineering or leaked documents required
  - Less work on the ’Layer 0’
  - Still, custom hardware in low quantity

A GSM baseband under our control

Alternative ’lazy’ approach

- Re-purpose existing mobile phone
  - Hardware is known to be working
  - No prototyping, hardware revisions, etc.
  - Reverse engineering required
  - Hardware drivers need to be written
  - But: More time to focus on the actual job: Protocol software
- Searching for suitable phones
  - As cheap as possible
  - Readily available: Many people can play with it
  - As old/simple as possible to keep complexity low
  - Baseband chipset with lots of leaked information

Baseband chips with leaked information

- Texas Instruments Calypso
  - DBB Documentation on cryptome.org and other sites
  - ABB Documentation on Chinese phone developer websites
  - Source code of GSM stack / drivers was on sf.net (tsm30 project)
  - End of life, no new phones with Calypso since about 2008
  - No cryptographic checks in bootloader
- Mediatek MT622x chipsets
  - Lots of Documentation on Chinese sites
  - SDK with binary-only GSM stack libraries on Chinese sites
  - 95 million produced/sold in Q1/2010

Initial choice: TI Calypso (GSM stack source available)

OsmocomBB Introduction

- Project was started only in January 2010 (9 months ago!)
- Implementing a GSM baseband software from scratch
- This includes
  - GSM MS-side protocol stack from Layer 1 through Layer 3
  - Hardware drivers for GSM Baseband chipset
  - Simple User Interface on the phone itself
  - Verbose User Interface on the PC
- Note about the strange project name
  - Osmocom = Open Source MObile COMmunication
  - BB = Base Band

OsmocomBB Software Architecture

- Reuse code from OpenBSC where possible (libosmocore)
    - We build libosmocore both for phone firmware and PC
- Initially run as little software in the phone
    - Debugging code on your host PC is so much easier
    - You have much more screen real-estate
    - Hardware drivers and Layer1 run in the phone
    - Layer2, 3 and actual phone application / MMI on PC
    - Later, L2 and L3 can me moved to the phone

OsmocomBB Software Interfaces

- Interface between Layer1 and Layer2 called L1CTL
  - Fully custom protocol as there is no standard
  - Implemented as message based protocol over Sercomm/HDLC/RS232
- Interface between Layer2 and Layer3 called RSLms
  - In the GSM network, Um Layer2 terminates at the BTS but is controlled by the BSC
  - Reuse this GSM 08.58 Radio Signalling Link
  - Extend it where needed for the MS case

OsmocomBB Target Firmware

- Firmware includes software like
  - Drivers for the Ti Calypso Digital Baseband (DBB)
  - Drivers for the Ti Iota TWL3025 Analog Baseband (ABB)
  - Drivers for the Ti Rita TRF6151 RF Transceiver
  - Drivers for the LCD/LCM of a number of phones
  - CFI flash driver for NOR flash
  - GSM Layer1 synchronous/asynchronous part
  - Sercomm - A HDLC based multiplexer for the RS232 to host PC

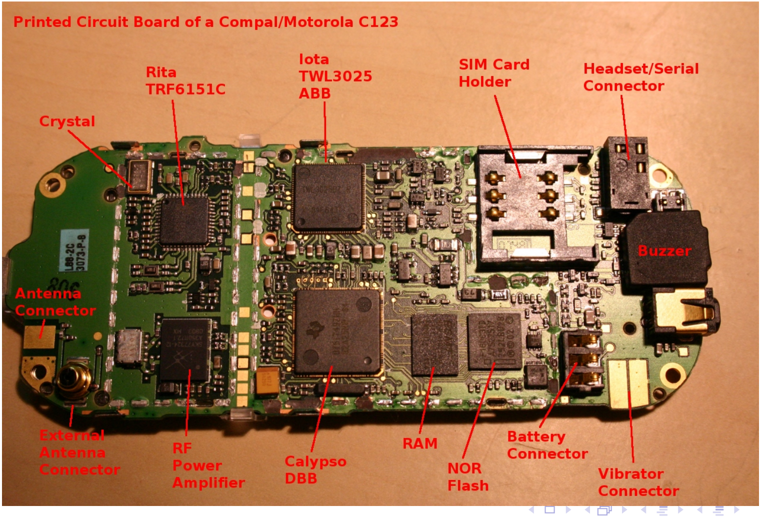
OsmocomBB Host Software

- Current working name: layer23
- Includes
  - Layer 1 Control (L1CTL) protocol API
  - GSM Layer2 implementation (LAPDm)
  - GSM Layer3 implementation (RR/MM/CC)
  - GSM Cell (re)selection
  - SIM Card emulation
  - Supports various 'apps' depending on purpose

OsmocomBB Supported Hardware

- Baseband Chipsets
  - TI Calypso/Iota/Rita
  - Some early research being done on Mediatek (MTK) MT622x
- Actual Phones
  - Compal/Motorola C11x, C12x, C13x, C14x and C15x models
  - Most development/testing on C123 and C155
  - GSM modem part of Openmoko Neo1973 and Freerunner
- All those phones are simple feature phones built on a ARM7TDMI based DBB

The Motorola/Compal C123



OsmocomBB Project Status: Working

- Hardware Drivers for Calypso/Iota/Rita very complete
- Drivers for Audio/Voice signal path
- Layer1
  - Power measurements
  - Carrier/bit/TDMA synchronization
  - Receive and transmit of normal bursts on SDCCH
  - Transmit of RACH bursts
  - Automatic Rx gain control (AGC)
  - Frequency Hopping
- Layer2 UI/SABM/UA frames and ABM mode
- Layer3 Messages for RR / MM / CC
- Cell (re)selection according GSM 03.22

OsmocomBB Project Status: Working (2/2)

OsmocomBB can now do GSM Voice calls (08/2010)

- Very Early Assignment + Late Assignment
- A3/A8 Authentication of SIM
- A5/1 + A5/2 Encryption
- Full Rate (FR) and Enhanced Full Rate (EFR) codec

OsmocomBB Project Status: Not working

- Fully-fledged SIM card reader inside phone (WIP)
- Layer1
    - Neighbor Cell Measurements
    - In-call hand-over to other cells
- Actual UI on the phone
- Circuit Switched Data (CSD) calls
- GPRS (packet data)

OsmocomBB Project Status: Executive Summary

- We can establish control/signalling channels to both hopping and non-hopping GSM cells
  - Control over synthesizer means we can even go to GSM-R band
- We can send arbitrary data on those control channels
  - RR messages to BSC
  - MM/CC messages to MSC
  - SMS messages to MSC/SMSC
- TCH (Traffic Channel) support for voice calls
  - Dieter Spaar and Andreas Eversberg have made multiple 20 minute call with current master branch
  - Some people have tried alpha code on real networks for real 30+ minute calls!

The wireshark protocol analyzer

- Software protocol analyzer for plethora of protocols
- Portable, works on most flavors of Unix and Windows
- Decode, display, search and filter packets with configurable level of detail
- Over 1000 protocol decoders
- Over 86000 display filters
- Live capturing from many different network media
- Import files from other capture programs
- Used to be called ethereal, but is now called wireshark
- `http://www.wireshark.org/`
- `http://www.wireshark.org/download/docs/user-guide-a4.pdf`

The wireshark protocol analyzer

GSM protocol dissectors in wireshark

- TCP/IP (transport layer for Abis/IP)
- E1 Layer 2 (LAPD)
- GSM Um Layer 2 (LAPDm)
- GSM Layer 3 (RR, MM, CC)
- A-bis Layer 3 (RSL)
  - A-bis OML for Siemens and ip.access in OpenBSC git
- GSMTAP pseudo-header (airprobe, OpenBTS, OsmocomBB)

wireshark integration in OsmocomBB

- OsmocomBB L1 runs on phone
- OsmocomBB L23 runs on host PC
- OsmocomBB L23 encapsulates 23byte L2 message in GSMTAP
- GSMTAP includes information not present in L2, such as
  - ARFCN, Timeslot
  - GSM Frame Number
  - Rx Signal Level / SNR
- OsmocomBB L23 sends GSMTAP message over UDP socket
- wireshark captures UDP packet like any UDP/IP

wireshark integration in OpenBTS and airprobe

- airprobe software runs on host PC
- implements Rx-only GSM L1 as SDR
- airprobe L23 encapsulates 23byte L2 message in GSMTAP
- wireshark captures UDP packet like any UDP/IP
- OpenBTS wireshark intergration similar, but for Rx + Tx

The wireshark protocol analyzer

Demonstration

Debugging SIM toolkit applications is hard

- Regular end-user phone does not give much debugging
- SIM card itself has no debug interface for printing error messages, warnings, etc.
- However, as SIM-ME interface is unencrypted, sniffing / tracing is possible
- Commercial / proprietary solutions exist, but are expensive

Introducing Osmocom SIMtrace

- Osmocom SIMtrace is a passive (U)SIM-ME communication sniffer
- Insert SIM adapter into actual phone
- Insert (U)SIM into SIMtrace hardware
- SIMtrace hardware provides USB interface to host PC
- `simtrace` program on PC encapsulates APDU in GSMTAP
- GSMTAP is sent via UDP to localhost
- wireshark dissector for GSM TS 11.11 decodes APDUs

Osmocom SIMtrace Hardware

- Hardware is based around AT91SAM7S controller
- SAM7S Offers two ISO 7816-3 compatible USARTs
- USARTs can be clock master (SIM reader) or slave (SIM card)
- Open Source Firmware on SAM7S implementing APDU sniffing
- Auto-bauding depending CLK signal, PPS supported
- Only prototype hardware right, but will be manufactured in Q1/2011

Part II - MTK and Free / Open Source Software

5 Open Source GSM tools for Debugging + Testing

6 Single-Core Android smart phone

7 Linux development and the community

Possible use cases for OpenBSC

OpenBSC or OpenBTS in R&D

- Inexpensive simulation of GSM network for R&D
- Flexible since any aspect can be modified by altering source code
- Complex and more exotic parts of GSM protocol spec can be tested
- Much more functionality than CMD 55 / Racal 6103 or similar
- Ability to send malformed L3 messages (fuzzing) for MTK MS stack security improvement

Possible use cases for airprobe

airprobe in R&D

- airprobe: Tracing of Um air interface
- SIMtrace: Tracing of SIM card interface

General advantages of FOSS based solution

- MTK has full access to source code
- New features can be added on any level of the protocol stack
- No dependency on a single supplier
- Lower cost means available to more MTK engineers
- Lower cost means available to more MTK customers (factory testing, field tests with OEM customers, ...)

MTK feature phone vs. smart phone

- MTK's advantage so far: Low cost sigle-core feature phone
  - Baseband processor runs Nucleus, GSM stack, UI and rich application stack (Camera, H.264, GPRS, TCP/IP, ...)
  - Other suppliers have to use dual core
- However, MTKs Nucleus based OS has custom/proprietary APIs
- Not many 3rd party applications can be installed on the phone
- Android, iPhone, Windows Mobile have standard API / environment
- Thus, MTK needs to offer 'standard' smart phone solution

Proposal: Single core Android smart phone

- Android, WinMobile, etc. have dual-core architecture
  - GSM/3G protocol stack on baseband processor
  - UI + applications on application processor
- If MTK now goes for Android smart phone, why go dual core?
  - Simply port L1 code into Linux kernel (IRQ/FIQ driven)
  - Make sure you follow the GPL and release L1 as Open Source
  - Run your L2/L3/L4 as proprietary userspace process on Linux
- Single-core Android phone has less ARM core licensing cost and less silicon size

SoC vendors and Linux ports

- A number of SoC vendors have been used with Linux for many years
- Port of Linux / BSP has originally been done by 3rd party or community
- SoC vendors started to become more active in the last 5 years
- Original: Create port, ship it to customer, done.
- SoC customers end up with vendor-specific code

Disadvantages of vendor ports

- Fast progress in mainline Linux kernel development
- Customers want latest kernel for latest features / performance
- Vendor port (not in mainline) always behind mainline
- Porting out-of-mainline vendor port into new mainline is lots of work
- Customers end up with old vendor-specific code

SoC vendors need to include their port mainline

- Major SoC vendors now work together with mainline developers
- Support SoC in latest mainline developer version
- Actively submit port into mainline Linux kernel
- Port in mainline stays automatically current/up-to-date
- Continued maintenance effort is shared by all parties

Further Reading

- Open source Software on a GSM protocol level

OpenBSC	`http://openbsc.osmocom.org/`
OpenBTS	`http://openbts.org/`
OsmocomBB	`http://bb.osmocom.org/`
airprobe	`http://airprobe.org/`

- A5 security related publications

A5 public `http://groups.google.com/group/uk.telecom/msg/ba76615fef32ba32`
Biham2003 `http://cryptome.org/gsm-crack-bbk.pdf`
Biham2006 `http://www.cs.technion.ac.il/users/wwwb/cgi-bin/tr-get.cgi/2006/CS/CS-2006-07.pdf`
HAR2009 `https://har2009.org/program/attachments/119_GSM.A51.Cracking.Nohl.pdf`
rainbow tables `http://reflextor.com/trac/a51/wiki`