SkyComm Mobile access for aviation industry

By

Aliaa Abdelhamid Mohamed

Hend Ahmed Mohamed omar

Fatma Mostafa Taha

Mennat Allah Mohamed Nasr

Mennat Allah Mohamed Hanfy

Nada Abdelgleel Sayed

Under supervision of

Dr. Hassan Mostafa Hassan

A Graduation Project Report Submitted to
the Faculty of Engineering at Cairo University
in Partial Fulfillment of the Requirements for the
Degree of
Bachelor of Science
in
Electronics and Communications Engineering

Faculty of Engineering, Cairo University

Giza, Egypt

July 2017

Table of Contents

List of Tables

List of Figures

List of Symbols and Abbreviations

ADC	Analog-to-Digital Converter
ADIRU	Air data inertial reference unit
AMPS	Advance Mobile Phone Service
AP	Access Point
ARFCN	Absolute radio frequency channel number
AUC	Authentication Center
BPU	Base Processing Unit
BSC	Base Station Controller
BSS	Base Station Subsystem
BSU	Beam-steering unit
BTS	Base Transceiver Station
DAC	Digital-to-Analog Converter
dBm	Decibel-mill watt
DDC	Digital Down Conversion
DLNA	Duplexer Low Noise Amplifier
DUC	Digital Up Conversion
EIR	Equipment Identity Register
ETSI	European Telecommunication Standards Institute
FCS	FemtoCell convergence server
FDMA	Frequency Division Multiple Access
FEC	Frequency Error Correction
FMPA	Flange Mounted Power Amplifier
FPGA	Field-Programmable Gate Array
FSK	Frequency Shift Key
FTTH	Fiber to the home
GEO	Geostationary orbit
GigE	Gigabit Ethernet
GMSK	Gaussian Minimum Shift Keying
GPRS	General Packet Radio Service
GPS	Global Positioning System

GSM	Global Systems for Mobile Communications
HGA	High Gain Antenna systems
HLR	Home location Register
HNBAP	Home Node B Application Part
HPA	High-power amplifiers
IAX	Inter-Asterisk exchange
IEEE	Institute of Electrical and Electronic Engineers
IETF	Internet Engineering Task Force
IF	Intermediate frequency
IMEI	International Mobile Equipment Identity
IMT-2000	International Mobile Telecommunications-2000
IP	Internet Protocol
IS-95	Interim Standard 95
ISI	Inter Symbol Interference
ISP	Internet service provider
kbps	kilobits per second
LEO	Low earth orbit
LNA	Low-noise amplifiers
LNB	Low-noise block down converter
MEO	medium earth orbit
MMS	Multimedia message service
MSC	Mobile Switching Center
MSISDN	Mobile Station International Subscriber Directory Number
MSK	Minimum Shift Keying
NDA	Non-Disclosure Agreement
NMT-450	Nordic Mobile Telephone-450
NMT-900	Nordic Mobile Telephone-900
NSS	Network Switching Subsystem
OMC	operation and maintenance center
OSS	Operation Support System
PSK	Phase Shift Keying
PSTN	Public Switched Telephone Network
QoS	Quality of service

QPSK	Quadrature Phase Shift Keying
RF	Radio Frequency
ROF	Radio over fiber
RTP	Real-Time Transport Protocol
RUA	RANAP User Adaptation
RX	Receiver
SCDMA	Synchronous Code Division Multiple Access
SDR	Software Defined Radio
SDU	Satellite data unit
SIM	subscriber identification module
SIP	Session Initiation Protocol
SMTP	Simple Mail Transfer Protocol
TACS	Total Access Communication System
TDMA	Time Division Multiple Access
TX	Transmitter
UDP	User Datagram Protocol
UHD	USRP Hardware driver
UMTS	Universal Mobile Telecommunications System
USRP	Universal Software Radio Peripheral
VLR	Visitor Location Register
VoIP	Voice over IP
VSAT	Very small aperture terminal
WAP	Wireless Application Protocol
W-CDMA	Wide band Code Division Multiple Access
WIMAX	Worldwide Interoperability for Microwave Access

Acknowledgments

This project would not have been possible without the support of many people. Many thanks to our advisers, Dr. Hassan Mustafa, Eng. Osama Mohamed and last year team who helped make some sense of the concept of the project and provided all their efforts to enable us complete the project and get best results. Thanks to Cairo University and our beloved department. And finally, thanks to our families, and numerous friends who endured this long process with us, always offering support and love.

Abstract

Providing network service to mobiles during flights in order to be connected to the network inside or outside the plane is very important nowadays. This solution has been a trend in many air flights companies such as Air France, Emirates and Etihad airways. It is also important to provide a movable network for remote places suffering from absence of coverage and institutions seeks to have private network. Sky communication helps in the execution of these networks in affordable prices providing different services such as phone calls, SMS and data in a safe way.

Chapter 1: Introduction

At present, means of communications become necessary part of life in human being life. It would be hard to imagine a world without means of communications like cell phones. As communication means plays a fundamental role in both personal life and professional life of individuals, it won't be acceptable to be isolated for hours on boarding or to be in a far area without coverage.

Sky communication targets to provide safe usage of mobile phones on board by providing slightly low coast network coverage. By using skycomm in planes you will be able to make phone calls, send and receive SMS and browse the internet. Aero mobile company provides these services using satellites, however a voice call costs in average 5$ per minute, it also provides data service but the speed is low. In this project we tried to overcome these problems seeking to provide those services in affordable prices for everyone. On the other hand, this idea will be applicable not only on board but it can be used also in any place with no coverage as it acts as a movable network so it can be used also to make a private network for a certain establishment.

Providing the service is simple, by using a software called "openbts" and a hardware called "USRP" we can establish a network service. The mobile access service starts on land but during taking off and landing it is automatically turned off by using a code in the USRP, it turns on again automatically at an altitude above 300 meter above the ground level. When the system is on, the passengers can switch on their mobiles and it will be no more airplane mode. Afterwards the passengers could turn on data roaming and choose the network name. There is a suggestion for ON AIR telecommunication mobile services companies that may help in having additional income by allowing other companies to have advertisement SMS’S. Finally, user could keep in touch at 30,000 feet.

In this book, in Chapter 2: we are providing a detailed solution for mobile access in aviation industry. We are explaining clearly how to establish a network. The starting point is to define the components and give total explanation of its role and how to use it.

In Chapter 3: we are giving cleared and detailed information about GSM and why we use it in addition to analogy between the traditional GSM network and the openbts GSM network.

The aim of Chapter 4: is to clarify and introduce SkyComm solution inside airplane in general way which will provide call flow starting from cell phone onboard till reach destination on earth. And describe each component in the solution in details. Which means providing methodology of our project.

Chapter 5: shows setup and steps of installation of our system.

Chapter 6: describes configurations of our system.

In Chapter 7: we will identify the different problems and solution that face us to make installation code and run USRP.

In Chapter 8: we are discussing all the ways to go outside the GSM network in order to communicate with outside world by using GSM gateway.

Chapter 9: All is about marketing and expensing of our idea and how to sell it in the aviation industry, also the system capacity is described in it.

In Chapter 10: we are providing future work and what we will be needed to upgrade our system.

Finally, in Chapter 11: we are providing our project and research results with brief conclusion.

Chapter 2: Requirements

2.1 Motivation

In this chapter, we will guide you through the requirements of hardware and software and we will identify all different requirements that face us to make our project.

2.2 OpenBTS (Open Base Transceiver Station) software:

OpenBTS is an open-source UNIX application that uses the Universal Software Radio Peripheral (USRP) to present a GSM air interface to standard GSM handset. OpenBTS implements most of the complexity involved in building a mobile network in software. It allows standard GSM-compatible mobile phones to make telephone calls without using existing telecommunication providers' networks by providing a way to create a low cost GSM cellular network. It has four main applications which enable it to replace the GSM Network Component:

2.2.1 OpenBTS:

It is the main application. It is responsible for implementing the GSM air interface in software and communicating directly with GSM handsets over it. This communication is converted into SIP and RTP on the IP network side and interacts with the components above to form the core network. It implements the GSM stack above the radio modem.

2.2.2 Asterisk:

Asterisk is an open source framework for building communications applications which turns an ordinary computer into a communications server. Asterisk powers IP PBX systems, VoIP gateways, conference servers and other custom solutions. It is free and open source which is sponsored by Digium. It became the basis for a complete business phone system, used to enhance or extend an existing system, or to bridge a gap between systems.

Asterisk is a real time voice and video applications as Apache is to web applications. It abstracts the complexities of communications protocols and technologies, allowing you to concentrate on creating innovative products and solutions.

You can use Asterisk to build communications applications, things like business phone systems (also known as PBXs), call distributors, VoIP gateways and conference bridges. Asterisk includes both low and high-level components that significantly simplify the process of building these complex applications.

But, why do we need to configure it?

We need to configure this application in order to route phone call between users correctly without any problems.

2.2.3 SIPAuthServe:

SIP Authorization Server (SIPAuthServe) is an application that processes SIP REGISTER requests that OpenBTS generates when a handset attempts to join the mobile network. SIPAuthServe is responsible for updating the subscriber registry database with the IP address of the OpenBTS instance that initiated it, allowing other subscribers to call the handset.

2.2.4 Smqueue:

SMQueue is an RFC-3428 store-and-forward server. It is used for text messaging in the OpenBTS system. It is required to send a text message from one mobile system to another, or to provide reliable delivery of text messages to a mobile system from any source.

There are other applications like transceiver application performs the radio modem functions of GSM and manages the USB interface to the radio hardware. It is responsible for transmitting and receiving samples to and from the USRP. It also performs the basic operations such as modulation, interleaving and correlation.

2.3 LINUX server:

One of the requirements is a standard commodity LINUX server. Other architectures are beginning to be supported, but stick to an x86 processor running a 32-bit operating system for the best results for now. This computer can be a separate machine in your test environment or it can actually be a virtual machine on the laptop or desktop you use daily.

2.4 USRP:

USRP is an abbreviation for Universal Software Radio Peripheral. It is a computer-hosted RF transceivers used for development and exploration of software-defined radios. USRP transceivers can transmit and receive radio-frequency signals in a several bands and can be used for applications in communications education and research. It allows standard GSM-compatible mobile phones to be used as SIP endpoints in Voice over IP (VOIP) networks used in many telecommunication usages especially in air craft. It is very small in size about a hand palm size. It is responsible on the RF domain, its main rule is to apply GSM signal through leaky feeder, considering the frequency band, the modulation techniques, number of channels, multiplexing technique, bandwidth, time slot duration The USRP product family is intended to be a comparatively inexpensive hardware platform for software radio, and is commonly used by research labs, universities, and hobbyists. And now we will identify two important USRP product family:

2.4.1 USRP B100

The Ettus Research USRP B100 as shown in figure 2-1 builds on the success of the USRP1 featuring a larger, more powerful FPGA, flexible sample rates, and onboard TCXO clock reference. This system connects to a host computer by USB 2.0 therefore providing up to 16 MHz of RF bandwidth. It contains a programmable FPGA, two 64 MS/s 12-bit ADCs, two 128 MS/s 14-bit DACs, and auxiliary analog and digital I/O to facilitate integration into a larger system. The USRP B100 can be synchronized to an external clock reference and pulse-per-second (PPS) signal.

Technical Features:

- 16 MHz of RF bandwidth with 8 bit samples
- 8 MHz of RF bandwidth with 16 bit samples
- USB 2.0 High Speed connectivity
- Motherboard has one RTX daughterboard slot (1 RX + 1 TX connectors)
- Onboard FPGA processing & FPGA: Xilinx Spartan 3A-1400 FPGA
- ADCs: 12-bits 64 MS/s & DACs: 14-bits 128 MS/s
- Ability to lock to external 5 or 10 MHz clock reference
- TCXO Frequency Reference (~2.5ppm) & Flexible clocking from 10 MHz to 64 MHz

Figure 2-1 Ettus Research USRP B100

2.4.2 USRP B200

The Ettus Research USRP B200 as shown in figure 2-2 provides a fully integrated, single board, Universal Software Radio Peripheral platform with continuous frequency coverage from 70 MHz –6 GHz. Designed for low-cost experimentation, it combines a fully integrated direct conversion transceiver providing up to 56MHz of real-time bandwidth, an open and reprogrammable Spartan6 FPGA, and fast and convenient bus-powered SuperSpeed USB 3.0 connectivity. Full support for the UHD (USRP Hardware Driver) software allows you to immediately begin developing with GNU Radio, prototype your own GSM base station with OpenBTS, and seamlessly transition code from the B200 to higher performance, industry ready USRP platforms. An enclosure accessory kit allows users of green PCB devices (revision 6 or later) to assemble a protective steel case.

Technical Features:

- The first fully integrated USRP device with continuous RF coverage from 70 MHz –6 GHz
- Full duplex operation with up to 56 MHz of real time bandwidth (61.44MS/s quadrature)
- Fast and convenient bus-powered connectivity using SuperSpeed USB 3.0
- GNU Radio and OpenBTS support through the open-source USRP Hardware Driver™ (UHD)
- Open and reconfigurable Spartan 6 XC6SLX75 FPGA with free Xilinx tools
- Early access prototyping platform for the Analog Devices AD9364 RFIC, a fully integrated direct conversion transceiver with mixed signal baseband

- Steel enclosure accessory kit available for green PCB devices (revision 6 or later)

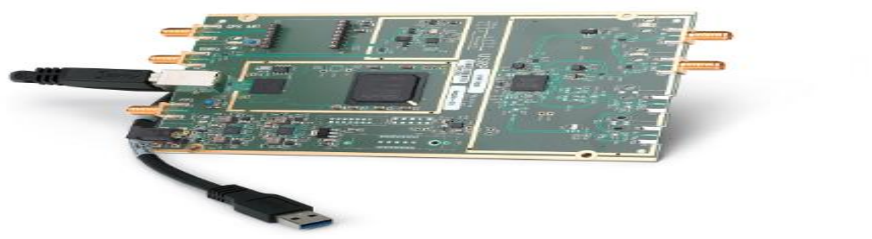


Figure 2-2 Ettus Research USRP B200

2.5 Laptop:

Laptop is responsible for all processing in the network.

2.6 Antennas:

Antennas are tuned for a specific frequency, so choose one that most closely matches the GSM band. The frequency also plays a role in the coverage area size. Low-frequency bands (850 and 900 MHz) propagate larger distances than high-frequency bands.as shown in figure2-3 we will use antenna vert900.

Figure 2-3 Antenna VERT900

2.7 VOIP GSM Gateway:

A VoIP GSM Gateway enables direct routing between IP, digital, analog and GSM networks. With these devices (fixed cellular terminals) companies can significantly reduce the money they spend on telephony, especially the money they spend on calls from IP to GSM. The core idea behind cost saving with VoIP GSM Gateways is Least Cost Routing (LCR). Through least cost routing the gateways select the most cost-effective telephone connection. They check the number which is dialed as well as rate information which is stored in an internal routing table. Because several SIM cards and GSM modules are integrated within the VOIP GSM Gateway it is able to make relatively cheaper GSM to GSM calls instead of expensive IP to GSM calls. In this project WGW1002G gateway is used.

WGE1002G gateway as shown in figure 2-4 has some advantages, it is described by easy-carrying and space-saving. The size of WGW1002G gateway is smaller than any GSM gateway and it is simple so it can make telecommunication more high quality and stable. The GSM gateway in a fixed configuration designs with a Lan Switch board that provides steady telecommunication. It supports SMS messages sending and receiving and group sending and SMS to email. The GSM Gateways use standard SIP protocol and compatible with Leading IMS/NGN platform, IPPBX and SIP servers, support most of the VoIP operating platforms such as Asterisk, Elastix, 3CX, Free SWITCH, Broad soft etc.

Technical Specifications:

- 2 GSM Channels & 1 SIM card per GSM channel
- 1 10/100M LAN interfaces
- 2 Standard SMA-K interfaces
- Storage temperature range: -20~70℃
- Operation temperature range: 0~40℃
- Operation humidity range: 10%~90% non-condensing
- Power source: 12V DC/2.33A & Power: 6W
- Dimension: 16cm*10.1cm*3.1cm &Weight: 237g

Figure 2-4 WGW1002G gateway

2.8 Test Phones:

For testing, at least two GSM handsets compatible with the band are needed.

Chapter 3: Global System for Mobile Communications

3.1 Introduction:

GSM is a standard developed by the European Telecommunications Standards Institute (ETSI) to describe the protocols for second-generation (2G) digital cellular networks used by mobile phones. GSM network is developed as a replacement for first generation (1G) analog cellular networks, and the GSM standard originally described as a digital, circuit-switched network optimized for full duplex (allow both callers to speak and be heard at the same time) voice telephony.

The second generation (2G) of the wireless mobile network was based on low-band digital data signaling. The most popular 2G wireless technology is known as Global Systems for Mobile Communications (GSM). The first GSM systems used a 25MHz frequency spectrum in the 900MHz band. Using FDMA (Frequency Division Multiple Access), which is a standard that lets multiple users access a group of radio frequency bands and eliminates interference of message traffic, is used to split the available 25MHz of bandwidth into 124 carrier frequencies of 200 kHz each. Each frequency is then divided using a TDMA (Time Division Multiple Access) scheme into eight time slots and allows eight simultaneous calls on the same frequency. This protocol allows large numbers of users to access one radio frequency by allocating time slots to multiple voice or data calls. TDMA breaks down data transmission, such as a phone conversation, into fragments and transmits each fragment in a short burst, assigning each fragment a time slot. With a cell phone, the caller does not detect this fragmentation.

Today, GSM systems operate in the 900 MHz and 1.8 GHz bands throughout the world with the exception of the Americas where they operate in the 1.9 GHz band. 2G wireless technologies can handle some data capabilities such as fax and short message service, but it is not suitable for web browsing and multimedia applications. So-called ‘2.5G’ systems recently introduced enhance the data capacity of GSM and mitigate some of its limitations.

3.2 GPRS:

GPRS is a second generation (2G) and third generation (3G) or sometimes referred to as in-between both generations, 2.5G--wireless data service that extends GSM data capabilities for Internet access. GPRS provides both a means to aggregate radio channels for higher data bandwidth and the additional servers required to off-load packet traffic from existing GSM circuits. It supplements today's Circuit Switched Data and Short Message Service. GPRS is not related to GPS (the Global Positioning System), a similar acronym that is often used in mobile contexts.

Theoretical maximum speeds of up to 171.2 kilobits per second (kbps) are achievable with GPRS using all eight time slots at the same time. This is about ten times as fast as current Circuit Switched Data services on GSM networks. However, it should be noted that it is unlikely that a network operator will allow all time slots to be used by a single GPRS user. Additionally, the initial GPRS terminals (phones or modems) are only supporting only one to four time slots. The bandwidth available to a GPRS user will therefore be limited.

3.3 Traditional GSM structure:

The GSM network architecture as shown in figure 3-1 consists of different elements that all interact together to form the overall GSM system. These include elements like the base-station, controller, MSC, AUC, HLR, VLR, etc.

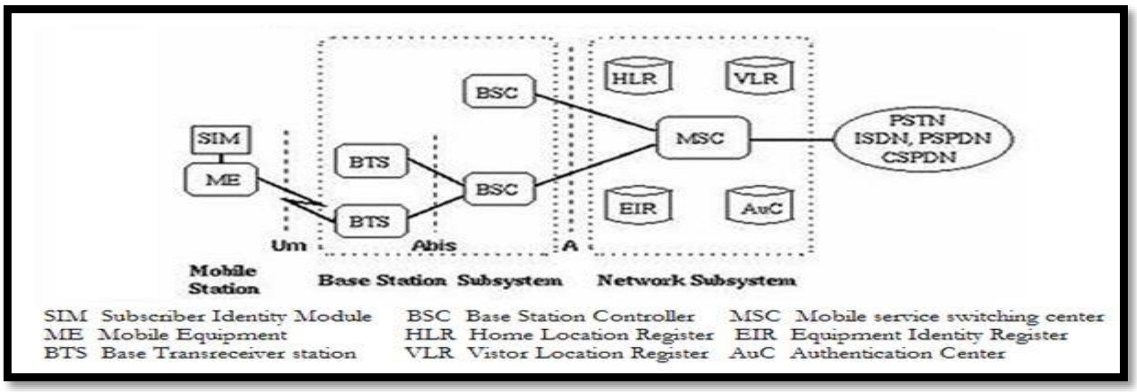

SIM
ME
BTS
BTS
BSC
BSC
HLR
VLR
MSC
EIR
AuC
PSTN
ISDN, PSPDN
CSPDN
Um
Abis
A
Mobile Station
Base Station Subsystem
Network Subsystem
SIM Subscriber Identity Module
ME Mobile Equipment
BTS Base Transreceiver station
BSC Base Station Controller
HLR Home Location Register
VLR Vistor Location Register
MSC Mobile service switching center
EIR Equipment Identity Register
AuC Authentication Center

Figure 3-1 GSM structure

The GSM network architecture as defined in the GSM specifications can be grouped into four main areas:

- Mobile station (MS)
- Base-Station Subsystem (BSS)
- Network and Switching Subsystem (NSS)
- Operation and Support Subsystem (OSS)

The different elements of the GSM network operate together within the system.

3.3.1 Mobile Station (MS):

The MS is made up of the mobile equipment and a SIM. The SIM enables the user to have access to subscribed services irrespective of a specific terminal. The insertion of the SIM card into any GSM terminal allows the user to receive calls on that terminal, to make calls from that terminal, and to use the other subscribed services. The ME is identified with an international mobile equipment identity (IMEI).

The SIM card contains, among other information, the international mobile subscriber identity (IMSI) used to identify the subscriber to the system, and a secret key for authentication. The IMEI and the IMSI are independent, thereby allowing personal mobility.

3.3.2 Base Station Subsystem (BSS):

The BSS is composed of several base station controllers (BSCs) and BTS. Each Base Transceiver Station defines a single cell.

3.3.2.1 Base transceivers station (BTS):

The BTS is the defining element for each cell. The BTS communicates with the mobiles and the interface between the two is known as the Um interface. The BTS contains the radio transceivers, responsible for the radio transmissions with the MS. This includes the following functions:

- Modulation and demodulation.
- coding and decoding.
- Encryption process.
- RF transmits and receives circuits (power control, management of antenna diversity, discontinuous transmission).

3.3.2.2 Base Station controller (BSC):

BSC controls a group of BTSs, and is often co-located with one of the BTSs in its group. It manages the radio resources and controls items such as handover within the group of BTSs. The interface that connects a BTS to a BSC is called the A-bis interface.

3.3.3 Network station subsystem (NSS):

The NSS is responsible for the network operation. It provides the link between the cellular network and the Public switched telecommunicates Networks (PSTN or Data Networks). The NSS controls handoffs between cells in different BSSs, authenticates user and validates their accounts, and includes functions for enabling worldwide roaming of mobile subscribers. In particular, the switching subsystem consists of:

- Mobile switch center (MSC)
- Home location register (HLR)
- Visitor location Register (VLR)
- Authentications center (AUC)
- Equipment Identity Register (EIR)
- Interworking Functions (IWF)

3.3.3.1 Mobile switch center (MSC):

The main element within the core network area of the overall GSM network architecture is the Mobile Switching Services Centre (MSC). It is responsible for the registration of the mobile, the call establishing and routing and the mobility management. The MSC performs the telephony switching functions to the system. It also controls calls to and from other telephony or data system.

3.3.3.2 Home location register (HLR)

The HLR is database software that handles the management of the mobile subscriber account. It contains data of all mobile subscribers registered in its area like subscriber address, service type, current locations, forwarding address, authentication/ciphering keys, and billings information.

3.3.3.3 Visitor location Register (VLR)

The VLR is temporary database software similar to the HLR identifying the mobile subscribers visiting inside the coverage area of an MSC. The VLR assigns a Temporary mobile subscriber Identity (TMSI) that is used to avoid using IMSI on the air. The visitor location register maintains information about mobile subscriber that is currently physically in the range covered by the switching center. When a mobile subscriber roams from one LA (Local Area) to another, current location is automatically updated in the VLR. When a mobile station roams into a new MSC area, if the old and new LA's are under the control of two different VLRs, the VLR connected to the MSC will request data about the mobile stations from the HLR. The entry on the old VLR is deleted and an entry is created in the new VLR by copying the database from the HLR.

3.3.3.4 Authentications center (AUC)

The AUC database holds different algorithms that are used for authentication and encryptions of the mobile subscribers that verify the mobile user's identity and ensure the confidentiality of each call. The AUC holds the authentication and encryption keys for all the subscribers in both the home and visitor location register.

3.3.3.5 Equipment Identity Register (EIR)

The EIR is another database that keeps the information about the identity of mobile equipment such the International Mobile Equipment Identity (IMEI) that reveals the details about the manufacturer, country of production, and device type. This information is used to prevent calls from being misused, to prevent unauthorized or defective MSs, to report stolen mobile phones or check if the mobile phone is operating according to the specification of its type.

3.3.3.6 Interworking Functions (IWF)

The tasks of an IWF are particularly to adapt transmission parameters and protocol conversions. The physical manifestations of an IWF may be through a modem which is activated by the MSC dependent on the bearer service and the destination network.

3.3.4 operation support subsystem OSS:

The OSS or operation support subsystem is an element within the overall GSM network architecture that is connected to components of the NSS and the BSC. It is used to control and monitor the overall GSM network and it is also used to control the traffic load of the BSS. It must be noted that as the number of BS increases with the scaling of the subscriber population some of the maintenance tasks are transferred to the BTS, allowing savings in the cost of ownership of the system.

3.4 analogy between traditional GSM and openbts GSM network:

Our main target is to have all functions of BTS, BSC and MSC collapsed in the OpenBTS box. As the BTS contains the radio transceiver as mentioned before so the USRP take the place of BTS. Asterisk software code plays the role of MSC/VLR in processing all the calls incoming to, or originating from subscribers visiting the given coverage area provided by the USRP's Antenna. Smqueue and Asterisk play the role of HLR in the traditional GSM network which is the main database of permanent subscriber information for a mobile network. It stores an IMSI for each subscriber, authentication key, subscriber status and the current location. SIP Authentication Server is responsible for SIP registration and authentication server, used to process location updating requests from OpenBTS and perform corresponding updates in the subscriber registry database.

3.5 Why GSM?

GSM is a good choice precisely although it is old. Everyone knows it works and 80% of the world's carriers are still using it. It's a proven technology that is well-suited to the target application. About the coast it is less coast than 3G as it is open source and it's specification is publicly available. It can also be considered an easier technique than the 3G as in 3G you need to know the spreading code and the spreading process at all is more complicated.

Chapter 4: SkyComm solution for aviation industry

4.1 Motivation

In this chapter we will talk about the mechanism of calling between two mobile phones one of them in the plane and the other is not and we will identify the different phases and problems that face us to make the two mobile phones are attached to each other and how calling process is done successfully.

4.2 Safety design

The first question that should be asked is: Does any component affect the plane safety, plane navigation or communication with towers during flight? Or in other words, do we make interference to plane's electronic components during calls?

We have to show and prove that everything will work well and the answer of the two previous questions must be no. The navigation devices frequencies (according to Jneuhaus organization):

- Aircraft (Air carrier and Private), Aeronautical enroute "136.975 MHz".
- Aircraft (Air carrier and Private), Airport control tower, Automatic weather observation "136.400-136.450 MHz".
- Aircraft (Air carrier and Private), Flight test "123.575 MHz".
- Aircraft (Air carrier and Private), Airport control tower, Aeronautical search and rescue "123.100 MHz".
- Radio navigation land test "108.000 MHz".
- Localizer "108.100-108.150 MHz".

In the US the civil aircraft communications band (118-137 MHz) generally uses 25 kHz spaced channels. As of 2010 aeronautical enroute and flight test stations may use 8.33 kHz spaced channels in the 121.4-123.6, 128.825-132.0 and 136.5-136.875 MHz ranges. Therefore, no interference or damage may occur as the frequency is very far from that's used in plane as the minimum frequency used in mobile communication is 876 MHZ.

Now we are ready to talk about the phases of the solution in details after making sure that we are in the safe side of plane safety.

4.3 Solution overview

In order to solve the problem of applying mobile network on the plane we think critically to get the block diagram of our system. Firstly -because of the flight level- there are no mobile signal coverage in the plane, so we need an on air transceiver to apply this mobile signal coverage.

The next step is processing those signals and controlling the network to apply switching between calls, this step may be called base processing unit (BPU) which plays the rule of the core network and also contains the data base of the users of the network for authentication issues. Laptop with good processor and memory can model this

As we reach a laptop in BPU, we can easily use the IP network as a gateway of the network, and then we can connect to the satellite using the satellite unit which is implemented on the plane to reach the satellite link through the SATCOMM terminal.

The following figure 4-1 shows the design flow of the SkyComm system and we are going to explain the procedure of the call in this chapter depending on this flow.

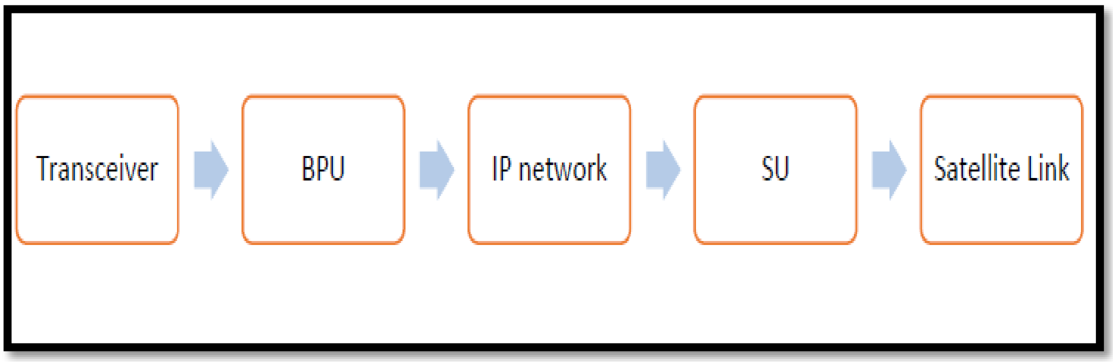


Figure 4-1Design flow of SkyComm

4.4 Call flow during flight

The following figure 4-2 shows the call flow during flight of the SkyComm system.

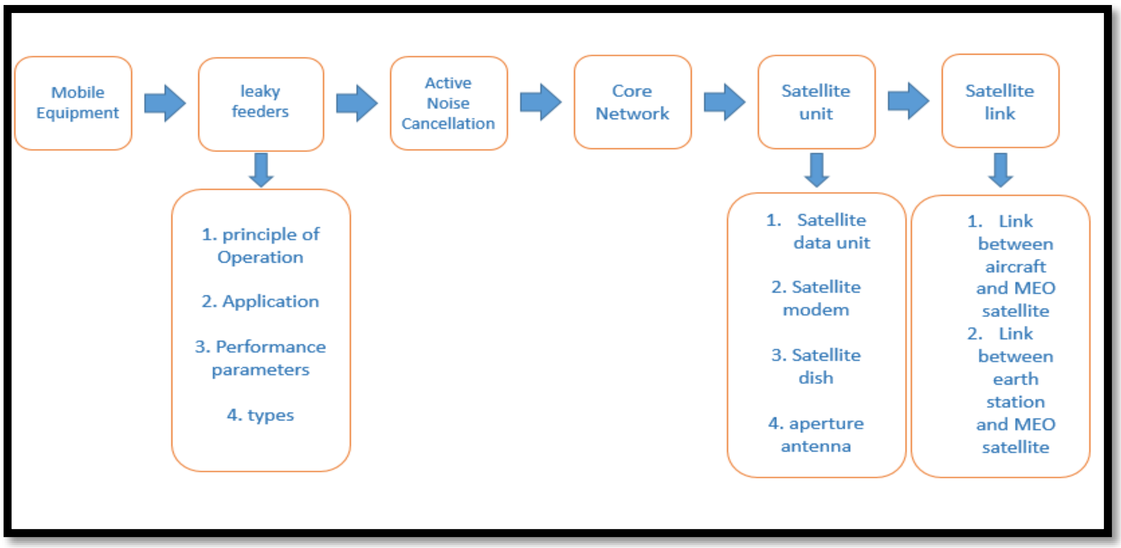


Figure 4-2 the call flow

4.4.1 Mobile Equipment

The mobile equipment has transceivers that can send and receive at 900, 1800, 1900, and 2100 MHZ these frequencies support the 2G and 3G families. We will also choose the 1800 MHZ as ETSI organization recommended as the minimum transmit power for a terminal in the 1800 MHZ band is lower than in 900 MHZ band (0 dBm instead of 5 dBm), and emissions at higher frequencies present higher path loss. The next step is the RF stage or antenna which supplies the signal to the mobile which will be selected to be leaky feeder for better coverage.

4.4.2 The leaky feeder

The leaky feeder is a coaxial cable that has small sections of its copper shielding stripped away to allow radio frequency (RF) signals to escape. Leaky feeders, which act as extended antennas, are also called radiating cables.

4.4.2.1 Principle of operation

A leaky feeder communication system consists of a coaxial cable run along tunnels which emits and receives radio waves, functioning as an extended antenna. The cable is "leaky" in that it has gaps or slots in its outer conductor to allow the radio signal to leak into or out of the cable along its entire length. Because of this leakage of signal, line amplifiers are required to be inserted at regular intervals, typically every 350 to 500 meters, to boost the signal back up to acceptable levels. The signal is usually picked up by portable transceivers carried by personnel. Transmissions from the transceivers are picked up by the feeder and carried to other parts of the tunnel, allowing two-way radio communication throughout the tunnel system.

The system has a limited range and because of the frequency it uses (typically VHF or UHF), transmissions cannot pass through solid rock, which limits the system to a line-of-sight application. It does, however, allow two-way mobile communication as duplexers used to allow the cable to carry other frequencies at the same time, so the same cable can provide extended mobile phone coverage with 900 and 1800 MHz signals being carried simultaneously.

4.4.2.2 Applications

Leaky feeders are used in places where the actual structure makes RF communication difficult. Leaky feeders are often used in structures with metal frameworks such as skyscrapers, tunnels, ships and planes to extend mobile coverage. They are also useful in situations where low power levels are required to prevent interference with wireless microphones or other communication technologies that share the same frequency spectrum.

a) In-flight wireless networks

Leaky feeder antenna system can also be used to allow reception of on-board GSM and Wi-Fi signals on passenger aircraft. The weight and space requirements of leaky feeder systems are usually lower than comparable antenna systems, thus saving space and fuel. The even field strengths produced by runs of leaky feeders spanning the entire fuselage improve coverage while requiring less transmitting power.

b) Mining

Leaky feeder has been used in the mining industry as a method of wireless communication between miners. The system is used as a primary communication system which has a transceiver small enough to be comfortably worn on a miner throughout an entire shift.

c) Underground railways

Leaky feeder system is also used for underground mobile communication in mass transit railways.

d) Industrial buildings

Leaky feeder is also being used in warehouses and other industrial buildings where it is difficult to get Wi-Fi coverage using normal access points. Real life installations with 50–75 meters of leaky wire connected to the antenna input of Access Points exist, and are working fine.

4.4.2.3 Basic performance parameters

The performance of a leaky feeder system may be characterized by two parameters:

a) Longitudinal attenuation

b) Coupling loss

The longitudinal attenuation is governed primarily by the factors which apply to normal transmission lines, such as construction, conductor size and dielectric. Additionally, there is a small loss component attributable to the leakage (or mode converters)

The coupling loss is, in general terms, the power loss between the feeder and a mobile antenna in its vicinity. For the commonly used coaxial types of leaky cable it is dependent on the degree of shielding in the feeder construction, the configuration of the shield or conductors and the permittivity of the dielectric. For a given cable construction, it should also be noted that the coupling loss is also dependent upon:

1. The environment in which the cable is mounted
2. The cable mounting position
3. The characteristics, position and orientation of the mobile antenna
4. The operating frequency.

4.4.2.4 Types of leaky feeders

a) Bifilar lines
b) Continuously leaky coaxial cables
c) Coaxial cables with periodic apertures
d) Cables with mode converters.

Types a) and b) are intrinsically non-radiating in the sense that a cable of infinite length extending in free space can only carry waves guided by the structure. However, any discontinuity along the cable causes mode conversion and radiation.

In type c), the periodic apertures are radiating discontinuities and act like elements of an antenna array. Maximum radiation is obtained in oblique directions determined basically by the ratio of the spatial interval to the wavelength.

In type d), the mode converters or radiating elements are separate discontinuities acting in isolation.

Example

GORE™ Leaky Feeder Antennas: Antennas provide consistent connectivity across a broad frequency range — from 400 megahertz up to 6 gigahertzes — making the antennas compatible with numerous communication standards.

Typical applications:

a) Wide-body aircraft.
b) Single-aisle aircraft.
c) Picocells for phone coverage.
d) Wi-Fi 802.11 a/b/g/n/ac and WiMAX.
e) Connectivity to Bluetooth, DECT, DECT2, Globalstar, GSM, IRIDIUM Sat, MMS, PDC, and TETRA protocols.

4.4.3 The ANC (Active Noise Cancellation)

The Active Noise Cancellation it is already located at mobile phones which generate a broadband noise floor which is being emitted through existing leaky line antenna masking reception, they measure and ensure that handsets can only connect to on board GSM network and will then operate with the lowest possible transmission power level GSM-1800 power control level its nominal output power of 0 dBm. This will result in significantly lower radiation levels than those experienced on average when using a mobile phone with terrestrial networks on ground. It is also used in safety from the interference of the airplane devices.

ANC is connected with the leaky line antenna, the Pico Cell or USRP and the server (if exists).

4.4.4 Core Network

The core network is the block which control and manage the network, switch between calls and containing the database of users to authenticate before establish calls.

For SkyComm solution the core network consists of USRP and laptop, USRP as an OpenBTS is a software-based GSM access point, allowing standard GSM-compatible mobile phones to be used as SIP endpoints in Voice over IP (VOIP) networks used in many telecommunication usages especially in air craft. It is very small in size about a hand palm size and the system uses Ethernet cabling for satellite linkage. It is available for most cellular technologies including GSM, CDMA, UMTS and LTE. It is responsible on the RF domain, its main rule is to apply GSM signal through leaky feeder; considering the frequency band, the modulation techniques, number of channels, multiplexing technique, bandwidth, time slot duration, etc.

Laptop is responsible for all processing in the network; it assigns channel and time slot to the users, takes decisions of assigned frequencies, power control, allocation and release of time channels, handover commands, synchronization, time advance, switching between users and call set-up procedure. From the other side it plays the rule of gateway of the network to the outside world through IP network by connecting to the satellite unit (SU). The detailed rule of USRP and OpenBTS is described before in chapter 2.

4.4.5 Satellite unit (SU)

Satellite unit consists of 4 components as following:

4.4.5.1 Satellite data unit:

A satellite data unit (SDU) is an avionics device installed in an aircraft that allows air/ground communication via a satellite network. It is an integral part of an aircraft's SATCOM (satellite communication) system. The device connects with a satellite via ordinary radio frequency (RF) communication and the satellite then connects to a ground station or vice versa. All satellite communication whether audio or data is processed by the SDU.

The SDU communicates with an on board MDDU (multi-purpose disk-drive unit) which maintains an updatable table of ground stations in the aircraft current area and the order of preference for selection of which ground station to use, which guides the choice of satellite. Along with analyzing data continuously sent from all ground stations (such as station status and the error rate of signals from each station) the SDU receives information about the aircraft's position and orientation from another on board system (ADIRU, air data inertial reference unit) which it passes to the BSU (beam-steering unit) to direct the signal beam from the aircraft to the chosen satellite.

The SDU complies with the latest ARINC 781 industry standard for Inmarsat SATCOM capability for classic Aeronautical, Swift64, and Swift-broadband operations. Even though the SDU is one of the lightest, smallest and most affordable SATCOM on the market, it is designed to maximize efficiency and scalability. The Satellite Data Unit (SDU) has a built-in amplifier or, depending upon installation constraints, it can be paired with an external Flange Mounted Power Amplifier (FMPA). The SDU is designed to work with the latest Inmarsat-approved High Gain Antenna systems (HGA) including a Duplexer Low Noise Amplifier (DLNA), and a Configuration Module (CM).

The fact that some companies provide the HGA-2100, DLNA-2100, and HCM-2100 ensures ease of installation of your complete SATCOM system, not to mention a single source for service when needed. Each Software Defined Radio (SDR) channel card performs failure reversion to the required cockpit safety (Classic Aero) function. The system is designed to maintain communications without pilot intervention during all flight phases. Handover between satellites and spot beams is accomplished automatically, based on preferences contained in the Secure ORT as defined in ARINC 781.

4.4.5.2 Satellite modem

A satellite modem or Sat Modem is a modem used to establish data transfers using a communications satellite as a relay. There is a wide range of satellite modems from cheap devices for home Internet access to expensive multi-functional equipment for enterprise use.

A "modem" stands for "modulator-demodulator". A satellite modem's main function is to transform an input bit stream to a radio signal and vice versa. There are some devices that include only a demodulator (and no modulator, thus only allowing data to be downloaded by satellite) that are also referred to as "satellite modems." These devices are used in satellite Internet access case uploaded data is transferred through a conventional PSTN modem or an ADSL modem).

4.4.5.3 Satellite dish

A satellite dish is a dish-shaped type of parabolic antenna designed to receive electromagnetic signals from satellites, which transmit data transmissions or broadcasts, such as satellite television.

Principle of operation:

The parabolic shape of a dish reflects the signal to the dish’s focal point. Mounted on brackets at the dish's focal point is a device called a feed horn. This feed horn is essentially the front-end of a waveguide that gathers the signals at or near the focal point and 'conducts' them to a low noise block down converter or LNB. The LNB converts the signals from electromagnetic or radio waves to electrical signals and shifts the signals from the down linked C-band and/or Ku-band to the L-band range. Direct broadcast satellite dishes use an LNBF, which integrates the feed horn with the LNB. (A new form of Omni directional satellite antenna, which does not use a directed parabolic dish and can be used on a mobile platform such as a vehicle was announced by the University of Waterloo in 2004.

The theoretical gain (directive gain) of a dish increases as the frequency increases. The actual gain depends on many factors including surface finish, accuracy of shape, feed horn matching. A typical value for a consumer type 60 cm satellite dish at 11.75 GHz is 37.50 db.

With lower frequencies, C-band for example, dish designers have a wider choice of materials. The large size of dish required for lower frequencies led to the dishes being constructed from metal mesh on a metal framework. At higher frequencies, mesh type designs are rarer though some designs have used a solid dish with perforations.

A common misconception is that the LNBF (low-noise block/feed horn), the device at the front of the dish, receives the signal directly from the atmosphere. For instance, on BBC News down link shows a "red signal" being received by the LNBF directly instead of being beamed to the dish, which because of its parabolic shape will collect the signal into a smaller area and deliver it to the LNBF.

Modern dishes intended for home television use are generally 43 cm (18 in) to 80 cm (31 in) in diameter, and are fixed in one position, for Ku-band reception from one orbital position. Prior to the existence of direct broadcast satellite services, home users would generally have a motorized C-band dish of up to 3 m in diameter for reception of channels from different satellites. Overly small dishes can still cause problems, however, including rain fade and interference from adjacent satellites.

4.4.5.4 Very small aperture antenna

A very small aperture terminal (VSAT) is a two-way satellite ground station with a dish antenna that is smaller than 3 meters. The majority of VSAT antennas range from 75 cm to 1.2 m. Data rates range from 4 kbit/s up to 16 Mbit/s. VSATs access satellites in geosynchronous orbit to relay data from small remote earth stations (terminals) to other terminals (in mesh topology) or master earth station "hubs" (in star topology).

VSATs are used to transmit narrow band data (e.g., point of sale transactions using credit cards, polling or RFID data, or SCADA), or broadband data (for the provision of satellite Internet access to remote locations, VOIP or video). VSATs are also used for transportable, on-the-move (utilizing phased array antennas) or mobile maritime communications. After passing through SU, the next step is reaching the satellite by a specified link.

4.4.6 Satellite Link

The only outlet of aeronautical communication is the satellite link, navigations, communication to ground stations or entrainment services such as internet or Aircomm services must pass through satellite link by SATCOM terminal which is fixed on the top of aircraft and connected to satellite data unit (SDU) with RF cable.

Satellite link is not ideal, there are some obstacles will come up against the sent data like fading, path loss and latency. Those obstacles must be studied well to be considered in the communication system to transmitter, to study those obstacles we need to study link properties of satellite from the aircraft represented in the terminal to the satellite then back from the satellite to the earth station and vice versa.

The first question must be asked which satellite orbit which serves the aeronautical communication services? Let's present a bit of peels about satellite communication system first then compare between the different types of orbits which can be used to be able to answer on this question.

The main utilize of satellite is for long distance communication because of the greater coverage area, independency on the distance and the higher bandwidth of satellite link but on the other hand it suffers from the large propagation delay.

One of the basic factor in satellite communication is the elevation angle which is the angle formed by the line of sight (The center of the satellite transmission beam) and the horizontal plane for an object above the horizontal plane. Elevation angle affects the satellite coverage area.

There are obstacles face the satellite communication, one of them is the attenuation which occur because atmosphere represented in rain and cloud and absorb some known frequencies, attenuation is also a function of elevation angle.

Now, we are ready to compare between the satellite orbits. There is no need to talk about the elliptical orbits; circular orbits are enough in case of aeronautical communication. There are three levels of circular orbits, Geostationary orbit (GEO), medium earth orbit (MEO) and low earth orbit (LEO).

GEO is in orbit around 36 Km above earth's surface and remain in the same position relative to the observer on surface of earth so in case of fixed object, the antenna will not track the satellite. But in case of aircraft the antenna must track the satellite but not rapidly because of the large coverage area and this reduces the Doppler's effect and handovers for the aircraft. On the other hand, it has large propagation delay and needs high transmit power.

LEO is much closer to the earth, ranging from 500 to 1500 Km above earth's surface. It's not at fixed position relative to the surface. It's better in signal strength and time delay but has lower coverage area and the aircraft will suffer from Doppler shifts and handovers especially with the high speed of the airplanes.

MEO is between 8000 and 18000 Km above the earth surface and it is tradeoff between GEO and LEO solutions in issues like coverage areas and handovers.

Actually the main three orbits GEO, MEO and LEO can serve the aircraft, but each one has special advantages and special problems. But generally the lower orbits can reduce system capacity limitations and latency for real-time communication when the higher orbits reduce the system cost and network complexity such as reducing the handovers and offer higher coverage area. Note that elevation angle must have minimum value, under this value the transmission can't be occurs due to the attenuation.

The most practical solution in our days is the MEO orbit because it offers good coverage, not bad latency and moderate system complexity and cost. In the following we will first study the link properties between the aircraft and MEO satellite then between earth station and MEO satellite.

To get the performance of the link we need to calculate the carrier to noise ratio, which differs between uplink and downlink. Carrier to noise ratio is dependent of the receiver gain, wavelength, saturation flux density and noise temperature, but what is the noise temperature in satellite communication?

Noise temperature is a kind of noise which found by the resistance of the satellite antenna which can establish noise power, the temperature noise depends on the temperature and bandwidth. It is calculated by value called figure of Merit which characterize the ratio between gains to total noise temperature.

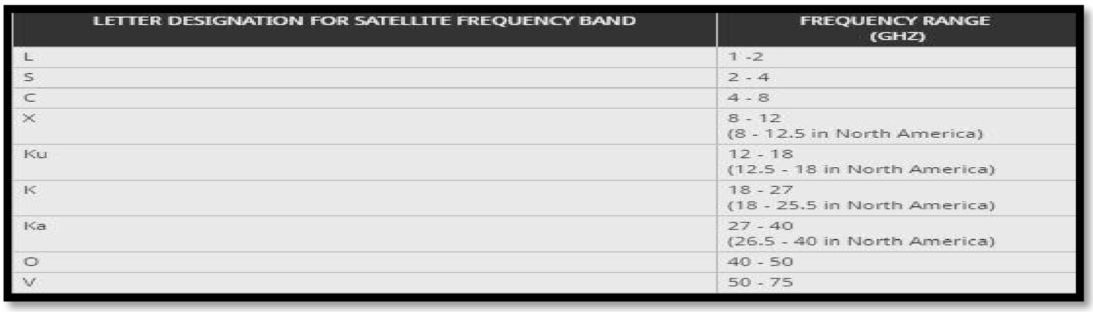
There are many other kinds of noise like Galactic noise which happens due to radiation from stars and planets and this kind of noise varies inversely with frequency, another kind is inter modulation noise which occurs where multiple carriers pass through any nonlinear device like travelling wave tube high power amplifier. Now we can take a pick of link properties in satellite communication with aircraft and earth station.

4.4.6.1 Link properties between the aircraft and MEO satellite:

The satellite link frequency between the aircraft and the satellite is at Ku band and above because the lower frequencies are too limited for multimedia applications. The following figure 4-3 shows the satellite frequency bands. The aeronautical channel has been investigated in K band at 18.685 GHz.

During normal flight the aeronautical channel has been reported to have constant power with small fading when antenna is in line of sight with satellite because there are no obstacles like buildings. Fading up to 13 dB occurs due to shadowing or diffraction from the aircraft structure especially at low elevation areas.

Signal may suffer also from attenuation due to atmosphere due to cloud, rains, vapor and oxygen. Atmospheric attenuation depends on the flight altitude, the region, and the weather conditions. The link properties for aircraft are not the same in different aircraft scenarios like parking, takeoff, landing and flight.



LETTER DESIGNATION FOR SATELLITE FREQUENCY BAND	FREQUENCY RANGE (GHZ)
L	1 -2
S	2 - 4
C	4 - 8
X	8 - 12 (8 - 12.5 in North America)
Ku	12 - 18 (12.5 - 18 in North America)
K	18 - 27 (18 - 25.5 in North America)
Ka	27 - 40 (26.5 - 40 in North America)
O	40 - 50
V	50 - 75

Figure 4-3the satellite frequency bands

4.4.6.2 Link properties between the Earth Station and MEO satellite:

The channel between the satellite and the earth station is different than the aeronautical channel because it is farther and may suffer from more obstacles for an example the path loss can completely change very quickly as a user moves from a clear state with a line of sight to a blocked state in a building or mountain but it doesn't suffer from handovers like the aeronautical one.

Satellite communication to earth has multipath effect and fading caused by buildings, hills and other obstacles on the ground. Other factors of link properties are path loss and attenuation which are increasing with the frequency and increasing with the elevation angle decreasing. Propagation impairments caused by the natural medium must be included in the characterization of the link channel between satellite and earth station, some of this impairments as mentioned in Suzuki model are:

1. Absorption of the atmosphere is lower than 1 dB and decreases as the elevation angle increases.
2. Attenuation caused by rain is proportional to the strength of rain but always less than 0.5 dB.
3. Attenuation caused by cloud or fog is less than 0.03 dB.
4. Attenuation caused by snow is less than 0.01 dB.
5. Reflection by atmosphere is less than 0.2 dB and happens when elevation angle is higher than 5 degrees.
6. Polarization effects make the loss reach 9 dB at specific frequencies.

4.4.7 How to reach mobile phone from Earth station?

A ground station, earth station, or earth terminal is a terrestrial radio station designed for extra planetary telecommunication with spacecraft (constituting part of the ground segment of the spacecraft system), or reception of radio waves from astronomical radio sources.

Ground stations may be located either on the surface of the Earth, or in its atmosphere. Earth stations communicate with spacecraft by transmitting and receiving radio waves in the super high frequency or extremely high frequency bands (e.g., microwaves). When a ground station successfully transmits radio waves to a spacecraft (or vice versa), it establishes a telecommunications link. A principal telecommunications device of the ground station is the parabolic antenna.

Earth station receives signal from satellite with very high frequency and need to convert signal to different destinations like mobiles, TV & and many applications. IN our subject we will deal with how to reach mobile phone?

Earth station after receiving the call there are many cabinets which have many roles one of them it's role how to reach mobile phones by using radio optical frequency (Rof) to reach public switched telephone network (PSTN) then go to gateway by means of communication and the call is treated as a normal call which we have talked about in chapter 2 (GSM) As shown in figure 4-4 below satellite uplink and downlink block diagrams.

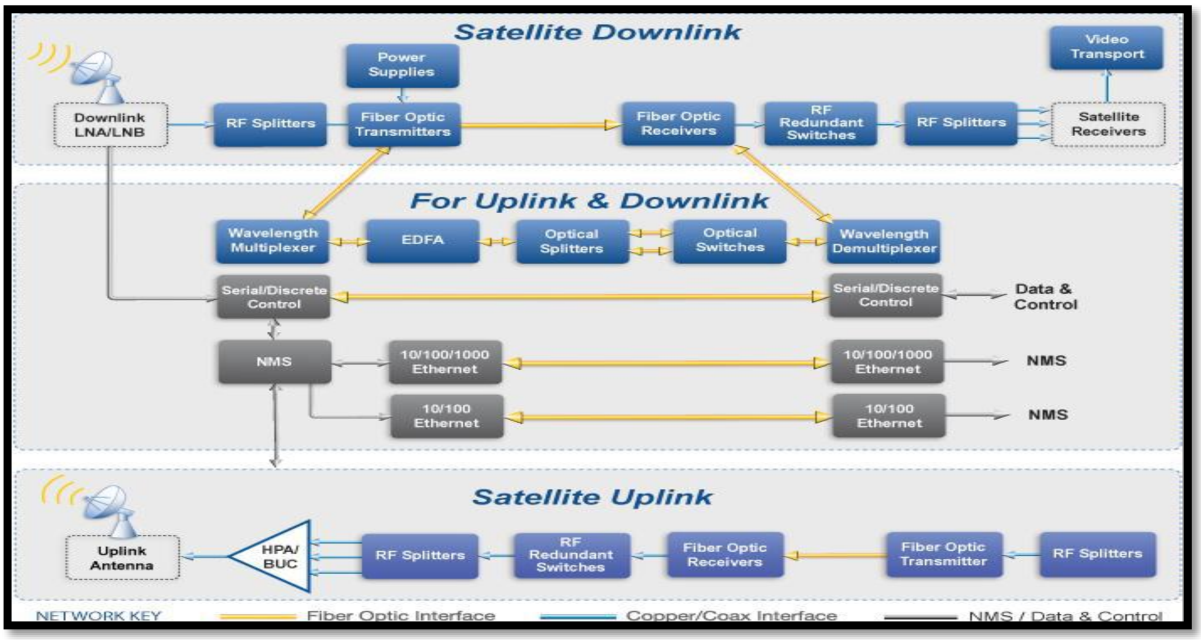


Figure 4-4 Uplink & downlink flow

Earth Station Components

An Earth terminal will always include an antenna. Terminals may also include:

1. Low-noise amplifiers (LNAs) or low-noise down-converters
2. High-power amplifiers (HPAs)

3. Signal processing equipment (e.g. down-converters, up-converters, IF amplifiers, modems and codecs)
4. Transmission and signaling equipment at the interface between the terminal and the terrestrial network
5. Supervisory and control equipment enclosures to protect the equipment from the environment

Note: Not everything is present in every terminal. For example, HPAs are not required for receive-only terminals.

Radio optical fibers

Radio over fiber (RoF) refers to a technology whereby light is modulated by a radio signal and transmitted over an optical fiber link to facilitate wireless access, such as 2G, 3G and Wi-Fi simultaneous from the same antenna. In other words, radio signals are carried over fiber-optic cable. Thus, a single antenna can receive any and all radio signals (2G, 3G, Wi-Fi, etc..) carried over a single-fiber cable to a central location where equipment then converts the signals; this is opposed to the traditional way where each protocol type (2G, 3G, Wi-Fi) requires separate equipment at the location of the antenna.

The Structure of Optical Fiber

As shown in figure 4-5 the structure of optical fiber.

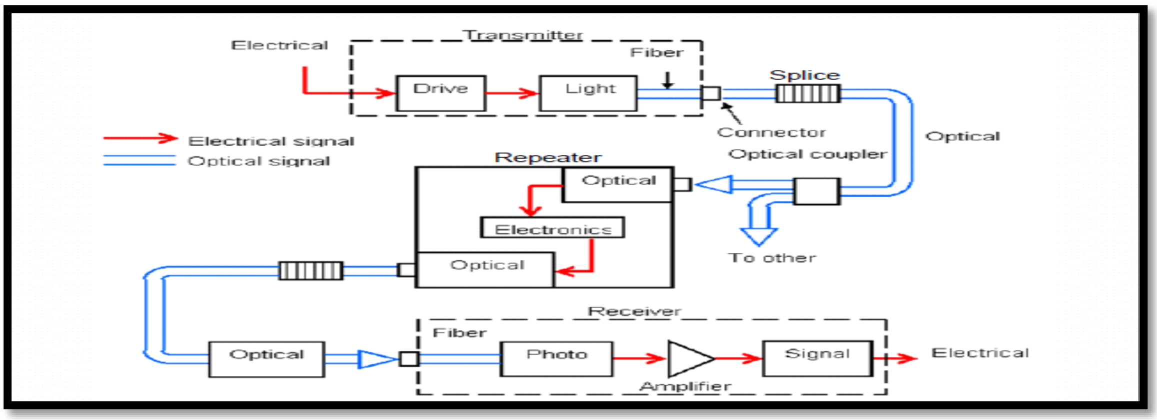


Figure 4-5 Optical Fiber Communication

Chapter 5: Installation

5.1 Motivation

In this chapter, we will guide you through the selection of hardware, installation of a base operating system, and development environment setup, as well as actually compiling and installing the components that compose the OpenBTS software suite.

5.2 For version openbts 4.0

This installation will be used for **USRP B100**.

5.2.1 Linux Server

Open the following web site and install Ubuntu release 12.04 then build it on your laptop.

```
$ http://releases.ubuntu.com/12.04/
```

5.2.2 Downloading the Code

The OpenBTS project consists of multiple software components, the several development scripts have been written to make it easier to download the code, switch branches, and compile components. To download these development scripts into your new environment, open the following web site and install release 4.0.0 source and binary:

```
$ http://openbts.org/get-the-code/
```

5.2.3 Installing Dependencies

If you're using a set of official release packages, you'll need to install some additional system libraries and define an additional repository source so all dependencies can be found and installed. Execute the following commands to define an additional repository source:

```
$ sudo apt-get -y install git-core autoconf automake  libtool g++ python-dev
  swig pkg-config libboost1.48-all-dev libfftw3-dev libcppunit-dev libgsl0-dev
  libusb-dev sdcc libsdl1.2-dev python-wxgtk2.8 python-numpy python-
  cheetah python-lxml doxygen python-qt4 python-qwt5-qt4 libxi-dev libqt4-
  opengl-dev libqwt5-qt4-dev libfontconfig1-dev libxrender-dev
```

```
$ sudo apt-get -y build-dep git-core autoconf automake  libtool g++ python-dev
  swig pkg-config libboost1.48-all-dev libfftw3-dev libcppunit-dev libgsl0-dev
```

```
libusb-dev sdcc libsdl1.2-dev python-wxgtk2.8 python-numpy python-cheetah python-lxml doxygen python-qt4 python-qwt5-qt4 libxi-dev libqt4-opengl-dev libqwt5-qt4-dev libfontconfig1-dev libxrender-dev
```

```
$ sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk2.8 git-core libqt4-dev python-numpy ccache python-opengl libgsl0-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq1 libzmq-dev python-requests python-sphinx
```

5.2.4 Git Compatibility

Git is a version-control system that manages software source code changes. The OpenBTS project utilizes several new features in Git, such as sub module branch tracking. To make sure your client is compatible (e.g., newer than 1.8.2), it needs to be updated.

First, execute this command to add support for Personal Package Archives, an alternate way to distribute binary release packages:

```
$ sudo apt-get install software-properties-common python-software-properties
```

Then, execute the following command to add a repository for the latest Git builds to your system:

```
$ sudo add-apt-repository ppa:git-core/ppa
```

Now, you must simply refresh the list of packages and install Git again to update your system's client:

```
$ sudo apt-get update
```

```
$ sudo apt-get install git
```

To confirm that the new Git client is installed properly, run the following command:

```
$ git --version
```

Now that you have Git installed, you can proceed to downloading the development scripts.

5.2.5 universal hardware driver (uhd)

The UHD is the universal hardware driver for Ettus Research products. The goal of the UHD is to provide a host driver and API for current and future Ettus Research products. It can be used standalone without GNU Radio.

Download the source tarballs. These commands will get the current version of uhd. Open new terminal.

```
$ sudo  git clone git://github.com/EttusResearch/uhd.git
```

open folder udh and create folder build:

```
$ cd ../

$ sudo cd uhd

$ sudo cd host

$ sudo mkdir build

$ sudo cd build
```

To Install uhd, then run with the following command:

```
$ sudo apt-get install cmake

$ sudo cmake ../

$ sudo make

$ sudo make test

$sudo make install
```

Congratulations! You now have uhd running on Ubuntu 12.04. To continue configuring uhd check out =chapter configuration.

5.2.6 A5/3 library

OpenBTS uses the A5/3 shared library to support call encryption. It contains cryptographic routines that must be distributed separately from OpenBTS:

Open new terminal then, from folder code, open liba53 by:

```
$ sudo cd liba53
```

To install liba53, run the following command:

```
$ sudo make install
```

5.2.7 Installing Components

By installing all of the following components on a fresh system, you are guaranteed a functional GSM network-in-a-box. Everything needed for voice, SMS, and data will be running in a single system.

The overall architecture of what you will be installing is visible in Figure5-1 . The Session Initiation Protocol (SIP) and Real-time Transport Protocol (RTP) are the two protocols that OpenBTS uses to convert GSM traffic into VoIP.

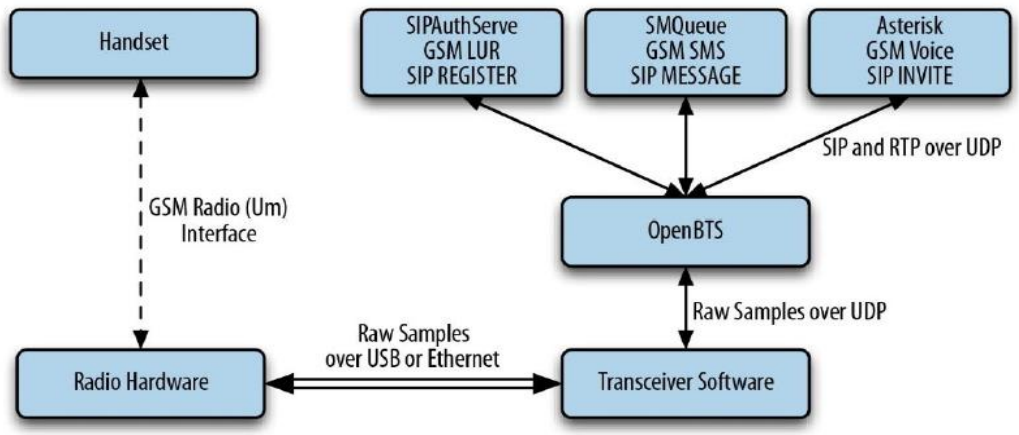

Handset
SIPAuthServe GSM LUR SIP REGISTER
SMQueue GSM SMS SIP MESSAGE
Asterisk GSM Voice SIP INVITE
SIP and RTP over UDP
GSM Radio (Um) Interface
OpenBTS
Raw Samples over UDP
Raw Samples over USB or Ethernet
Radio Hardware
Transceiver Software

Figure 5-1 Component architecture

5.2.8 OpenBTS

OpenBTS is responsible for implementing the GSM air interface in software and communicating directly with GSM handsets over it. This communication is converted into SIP and RTP on the IP network side and interacts with the components above to form the core network.

The GSM handsets see a fully compatible GSM radio access network and the core network sees standard SIP endpoints. Neither side must know that there is a layer between allowing the handsets to connect seamlessly to the IP world:

```
$ sudo cd openbts
```

to install openbts, then run with the following command:

```
$ sudo autoreconf -i

$ sudo ./configure --with-uhd

$ sudo make

$ sudo make install
```

With OpenBTS built, you now need to configure it to run correctly. There are a two key files that must be created for this to happen.

```
$ sudo mkdir /etc/OpenBTS

$ sudo sqlite3 -init ./apps/OpenBTS.example.sql /etc/OpenBTS/OpenBTS.db ".quit"
```

Create a symbolic link to the built Transceiver52M executable.

```
$ sudo cd apps

$ sudo ln -s ../Transceiver52M/transceiver .
```

At this point, we should be able to perform a basic sanity check of OpenBTS.

```
$ sudo ./OpenBTS
```

You should see output like this:

```
[system ready use the OpenBTSCLI utility to access CLI]
```

Congratulations! You now have openbts running on Ubuntu 12.04. To continue configuring openbts check out =chapter configuration

5.2.9 Asterisk

Asterisk is a VoIP switch responsible for handling SIP INVITE requests, establishing the individual legs of the call, and connecting them together. For this install I am using Asterisk 11.25.1 and will be compiling from source on Ubuntu 12.04.1.

Before you begin the install process you will want to be sure that your server OS is up to date.

```
sudo apt-get update

sudo apt-get upgrade -y
```

Next you will want to resolve basic dependencies.

```
$ sudo apt-get install build-essential wget libssl-dev libncurses5-dev libnewt-dev  libxml2-dev linux-headers-$(uname -r) libsqlite3-dev uuid-dev
```

Download the source tarballs. These commands will get the current version of Asterisk.

```
$sudo wget http://downloads.asterisk.org/pub/telephony/asterisk/asterisk-11-current.tar.gz
```

Extract the file from the tarballs

```
$ sudo tar zxvf asterisk*
```

Open folder asterisk and Install Asterisk.

```
$ sudo cd asterisk 11.25.1

$ sudo ./configure
```

Select your options when the menuselect command runs. Then select “Save & Exit” and the install will continue

```
$ sudo make menuselect  (save and exit)

$ sudo  make

$ sudo make install

$ sudo make config

$ sudo make samples
```

Start Asterisk and open new terminal and connect to the CLI

```
$ sudo asterisk –rvvvv
```

Congratulations! You now have Asterisk 11.25.1 running on Ubuntu 12.04. To continue configuring Asterisk check out =chapter configuration

5.2.10 SIPAuthServe

SIP Authorization Server (SIPAuthServe) is an application that processes SIP REGISTER requests that OpenBTS generates when a handset attempts to join the mobile network. When a handset authenticates successfully, SIPAuthServe is responsible for updating the subscriber registry database with the IP address of the OpenBTS instance that initiated it, allowing other subscribers to call the handset:

To setup the Subscriber Registry database you must first create the file path the db will reside in. By default, this is /var/lib/asterisk/sqlite3dir.

```
$ sudo mkdir -p /var/lib/asterisk/sqlite3dir
```

from folder code, open subscriberRegistry by:

```
$ sudo cd subscriberRegistry
```

to install subscriberRegistry:

```
$ sudo make
$ sudo make install
```

As with OpenBTS, you'll need to configure sipauthserve. We assume /etc/OpenBTS/ already exists.

```
$ sudo apt-get -y install sqlite3

$ sudo sqlite3 -init subscriberRegistry.example.sql
/etc/OpenBTS/sipauthserve.db ".quit"
```

Running sipauthserve will provide you with a registration server. To do so:

```
$ sudo ./sipauthserve
```

sipauthserve does not have a CLI, so you'll only see a small output:

```
[ ALERT 139639310980928 sipauthserve.cpp:214:main: ./sipauthserve (re)starting ]
```

Congratulations! You now have sipauthserve running on Ubuntu 12.04. To continue configuring sipauthserve check out =chapter configuration

5.2.11 SMQueue

SIP MESSAGE Queue (SMQueue) is an application that processes SIP MESSAGE requests that OpenBTS generates when a handset sends an SMS. It stores the messages, schedules them for delivery in the network, and reschedules them if the target handset is unavailable:

Open new terminal then, from folder code, open Smqueue by:

```
$ sudo cd smqueue
```

to install smqueue, then run with the following command:

```
$ sudo autoreconf -i

$ sudo ./configure

$ sudo make

$ sudo make install
```

As with OpenBTS, you'll need to configure smqueue. We assume /etc/OpenBTS/ already exists.

```
$ sudo sqlite3 -init smqueue/smqueue.example.sql /etc/OpenBTS/smqueue.db
".quit"
```

That will initialize /etc/OpenBTS/smqueue.db with default values. These configuration variables should work without modification, and are listed here.

Smqueue is run with the following command:

```
$ sudo ./smqueue
```

So you'll only see a small output:

```
[ ALERT 140545832068928 smqueue.cpp:2421:main: smqueue (re)starting ]
```

smqueue logs to syslogd facility LOCAL7, so there's not much to see here

Congratulations! You now have smqueue running on Ubuntu 12.04. To continue configuring smqueue check out =chapter configuration

the components that compose the OpenBTS software suite.

5.3 For version openbts 5.0

This installation will be used for **USRP B200.**

5.3.1 Linux Server

open the following web site and install Ubuntu release 14.04 then build it on your laptop.

```
$ http://releases.ubuntu.com/14.04.5/
```

5.3.2 Downloading the Code

The OpenBTS project consists of multiple software components, To download these development scripts into your new environment, open the following web site and install release 5.0.0 source and binary:

```
$ http://openbts.org/get-the-code/
```

5.3.3 Installing Dependencies

We will do the same commands, As in section 5.2.3

5.3.4 Git Compatibility

We will do the same commands, As in section 5.2.4**.**

5.3.5 universal hardware driver (uhd)

We will do the same commands, As in section 5.2.5.

5.3.6 Installing Components

Now, to download all of the components, simply run the clone.sh script:

```
$ cd dev
```

```
$ ./clone.sh
```

Each component's repository will be cloned from GitHub into your development environment.

The clone.sh script also automatically initializes any submodules needed.

Now that the OpenBTS project sources are in your development environment, you can

select a specific branch or release to compile. The switchto.sh script is used to toggle

between build version targets. For example, if you wanted To target the latest, greatest code in 5.0, run

the following command:

```
$ ./switchto.sh 5.0
```

The current version target can be listed for each component by using the state.sh

script. This script also lists any outstanding local changes for each component.

```
$ ./state.sh
```

5.3.7 Building the Code

Your development environment is now prepared to build the newest bits in the 5.0 series branch. To compile binary packages, you will use the build.sh script. It automatically installs the compiler and auto configuration tools as well as any required dependencies. It also controls which radio transceiver application will be built. As there are several different drivers available for the various radio types, build.sh requires an argument so it knows which hardware is being targeted (valid radio types are SDR1, USRP1, B100, B110, B200, B210, N200, and N210).

This book targets the Ettus Research N200. Run the build command now:

```
$ ./build.sh B200
```

This process can take a while (30–60 minutes) the first time it is run, depending on the hardware it is being executed on. Going forward, you will only need to recompile updated components. The following command, for example, recompiles just the OpenBTS package for an Ettus Research B200 radio:

```
$ ./build.sh B200 openbts
```

When the build script finishes, you will have a new directory named "BUILDS" containing a subdirectory with the build's timestamp. An example listing of this directory follows:

```
$ ls dev/BUILDS/2017-07-01--20-44-51/*.deb

liba53_0.1_i386.deb range-asterisk-config_5.0_all.deb

libcoredumper1_1.2.1-1_i386.deb range-configs_5.0_all.deb

libcoredumper-dev_1.2.1-1_i386.deb sipauthserve_5.0_i386.deb

openbts_5.0_i386.deb smqueue_5.0_i386.deb

range-asterisk_11.7.0.4_i386.deb
```

Congratulations! You can now move on to installing and starting each component, as well as learning what purpose each serves.

5.3.8 Installation

Now that you've downloaded a set of official release packages or compiled your own, they need to be installed and started. A bit of background for each component will be provided, followed by the installation procedure and any initializing configuration needed. As some components depend on others, they will be presented in the order needed to satisfy these interdependencies. Change into your new build directory before continuing:

```
$ cd dev/BUILDS/2014-07-29--20-44-51/
```

5.3.9 Coredumper library

OpenBTS uses the coredumper shared library to produce meaningful debugging information if OpenBTS crashes. Google originally wrote it and there are actually two libcoredumper packages: libcoredumper-dev contains development files needed to compile programs that utilize the coredumper library, and libcoredumper contains the shared library that applications load at runtime:

```
$ sudo dpkg -i libcoredumper1_1.2.1-1_i386.deb
```

5.3.10 A5/3 library

OpenBTS uses the A5/3 shared library to support call encryption. It contains cryptographic

routines that must be distributed separately from OpenBTS:

```
$ sudo dpkg -i liba53_0.1_i386.deb
```

5.3.11 System configs

This package contains a set of default configurations that will allow a fresh Ubuntu system to work out of the box when installed. It includes settings for the network interface, firewall rules, domain name system (DNS) configuration, logging, etc. You may not want to install this package if you are already comfortable configuring a Linux distribution, but its contents can serve as a guide for the required changes. During installation, you will be prompted several times to confirm the overwriting of certain configuration files. If you are unsure what the file does, a safe answer is always “Y” when dealing with a fresh system:

```
$ sudo dpkg -i range-configs_5.0_all.deb
```

5.3.12 Asterisk

Asterisk is a VoIP switch responsible for handling SIP INVITE requests, establishing the individual legs of the call, and connecting them together. There are two packages responsible for setting up an Asterisk installation that works without any additional configuration: range-asterisk and range-asterisk-configs.

The range-asterisk package contains a confirmed-working version of the Asterisk SIP switch software and ensures that the appropriate modules needed for OpenBTS are already included. No other patches to Asterisk are included; it is simply intended to represent the latest confirmed-working version of Asterisk.

The range-asterisk-configs package contains a set of configuration files so Asterisk knows about and can communicate with the subscriber registry database. This database is where the various components store and update subscribers’ phone numbers, identities, authentications, caller IDs, and registration states. Also, by using this database, it is no longer necessary to manually edit Asterisk configuration files when adding new handsets to the network:

```
$ sudo dpkg -i range-asterisk*.deb

$ sudo apt-get install –f
```

5.3.13 SIPAuthServe

```
$ sudo dpkg -i sipauthserve_5.0_i386.deb

$ sudo apt-get install -f
```

5.3.14 SMQueue

```
$ sudo dpkg -i smqueue_5.0_i386.deb
$ sudo apt-get install -f
```

5.3.15 OpenBTS

```
$ sudo dpkg -i openbts_5.0_i386.deb

$ sudo apt-get install -f
```

5.3.16 Starting/Stopping Components

Now that each component has been installed, you need to start them. Components are controlled on Ubuntu with a system named Upstart. Future releases of the OpenBTS suite will support other mechanisms such as systemd, but for now, Upstart is used. To start all components, execute the following:

```
$ sudo start asterisk

$ sudo start sipauthserve

$ sudo start smqueue

$ sudo start openbts
```

Conversely, to stop all components, use:

```
$ sudo stop openbts

$ sudo stop asterisk

$ sudo stop sipauthserve

$ sudo stop smqueue
```

Chapter 6: Initial Testing and Configuration

6.1 Motivation

The software and hardware should now be in place. This chapter will guide you through some initial sanity checks, functional testing, and basic configuration customization. By the end of this chapter, you will have successfully exchanged the first SMS messages and voice calls among phones over your private mobile network.

6.2 Initial State

Some of the manual steps that follow will conflict and fail if other instances of the services are already running. To make sure that nothing else is running on this system, execute the following:

```
$ sudo stop openbts

$ sudo stop asterisk

$ sudo stop sipauthserve

$ sudo stop smqueue
```

Now you can proceed to confirm connectivity at each step in the chain before running the first basic tests.

6.3 Confirm Radio Connectivity

The first thing you should verify is that the transceiver application can communicate with the radio hardware. Different vendors have different methods for accomplishing this.

6.3.1 Ettus Research Radios

All Ettus hardware uses the Transceiver52M binary, which was installed in ***/OpenBTS*** in the last chapter.

Run it as follows to see if the hardware device is detected:

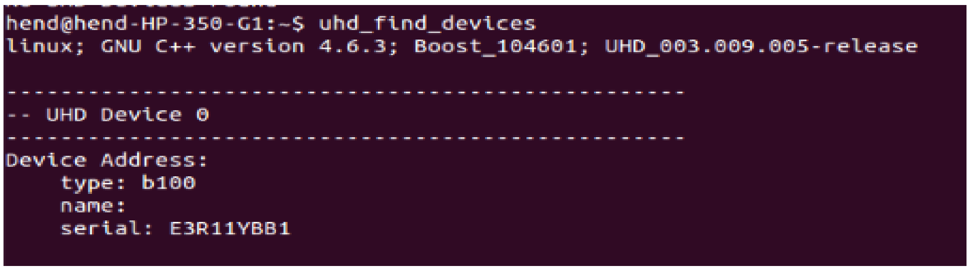
```
$ cd /OpenBTS

$ sudo ./transceiver

[sudo] password for openbts:

linux; GNU C++ version 4.8.4; Boost_105400; UHD_003.010.git-

119-g42a3eeb6 Using internal clock reference

-- Opening a USRP2/N-Series device...

-- Current recv frame size: 1472 bytes

-- Current send frame size: 1472 bytes
```

The example above shows a successful attempt. The transceiver can be stopped by pressing

Ctrl-C. If you instead see something like the following output, there is a problem:

```
$ cd /OpenBTS

$ sudo ./transceiver

[sudo] password for openbts:

linux; GNU C++ version 4.6.3; Boost_104601;

UHD_003.007.002-release Using internal clock reference

ALERT 1745:1745 2014-09-05T22:37:00.4 UHDDevice.cpp:528:open: No

UHD devices found with address ''

ALERT 1745:1745 2014-09-05T22:37:00.4 runTransceiver.cpp:160:main:
Transceiver

exiting...
```

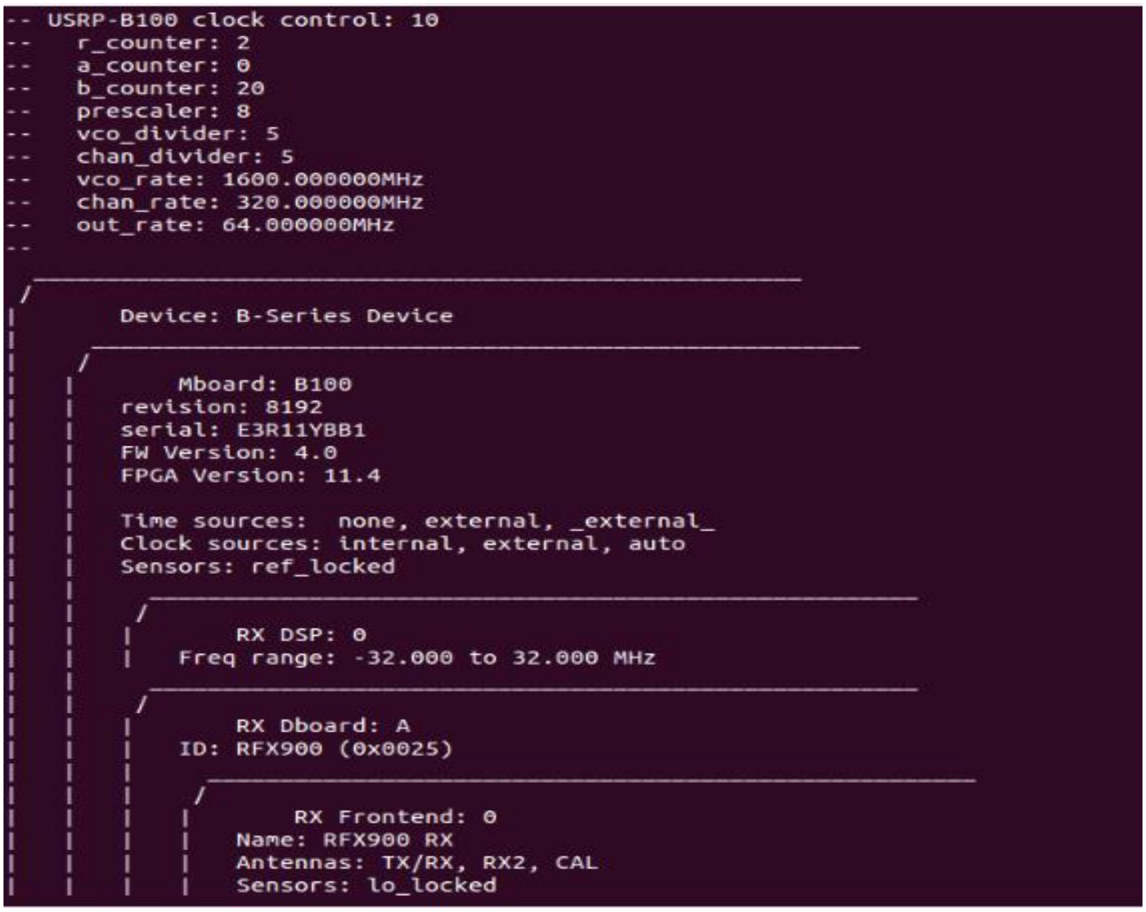
Ettus provides a couple of helper applications to automatically detect and inspect attached radios. Run the following command to list all attached devices. This example shows a B200 via USB:



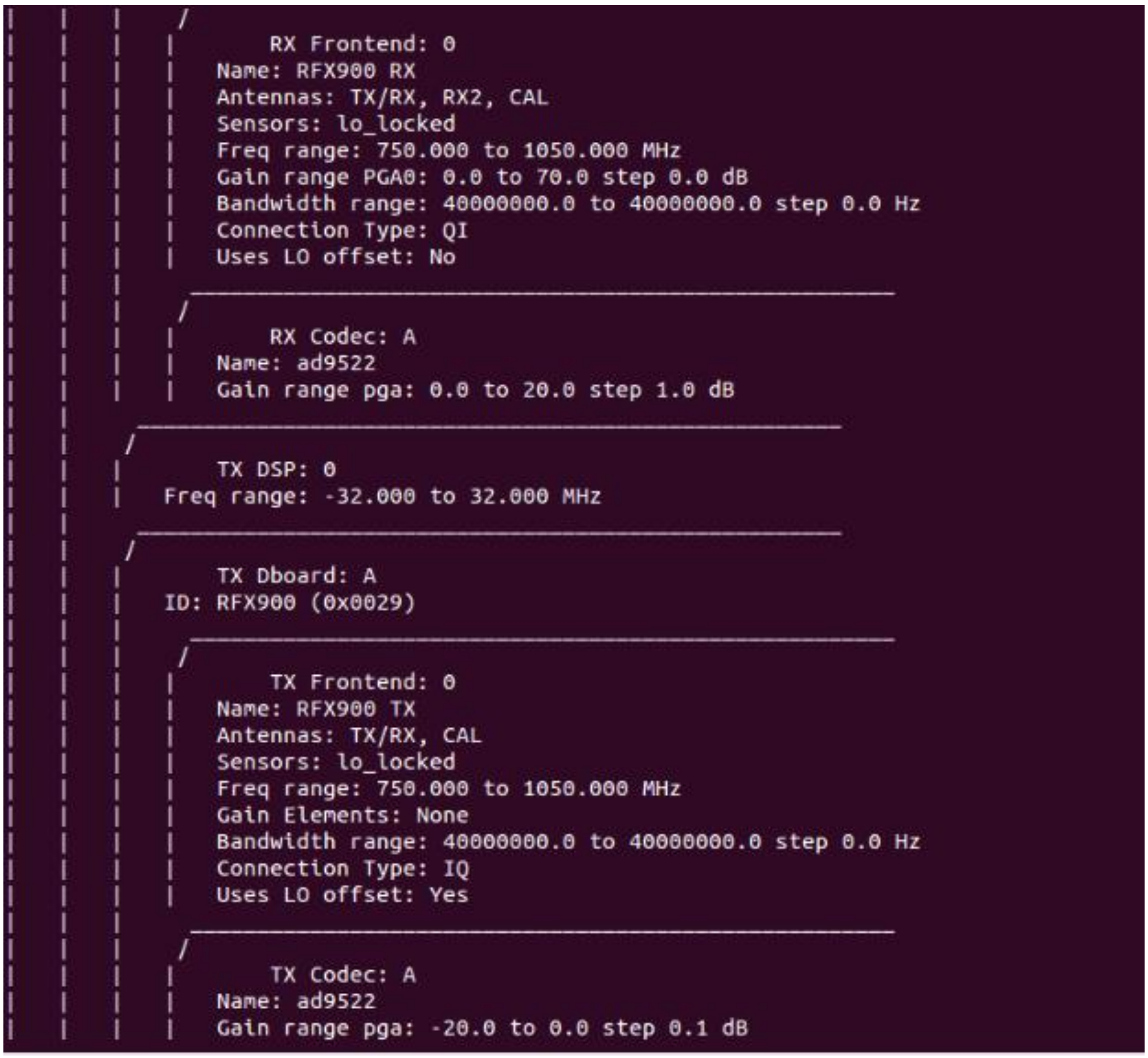
```
hend@hend-HP-350-G1:~$ uhd_find_devices
linux; GNU C++ version 4.6.3; Boost_104601; UHD_003.009.005-release

--------------------------------------------------
-- UHD Device 0
--------------------------------------------------
Device Address:
    type: b100
    name:
    serial: E3R11YBB1
```

Another helpful application is uhd_usrp_probe, which will inspect a device and return its technical information and configuration. An example run of this application follows:

$uhd_usrp_probe



```
-- USRP-B100 clock control: 10
--    r_counter: 2
--    a_counter: 0
--    b_counter: 20
--    prescaler: 8
--    vco_divider: 5
--    chan_divider: 5
--    vco_rate: 1600.000000MHz
--    chan_rate: 320.000000MHz
--    out_rate: 64.000000MHz
--
  _____________________________________________________
 /
|       Device: B-Series Device
|     _____________________________________________________
|    /
|   |       Mboard: B100
|   |   revision: 8192
|   |   serial: E3R11YBB1
|   |   FW Version: 4.0
|   |   FPGA Version: 11.4
|   |
|   |   Time sources:  none, external, _external_
|   |   Clock sources: internal, external, auto
|   |   Sensors: ref_locked
|   |     _____________________________________________________
|   |    /
|   |   |       RX DSP: 0
|   |   |   Freq range: -32.000 to 32.000 MHz
|   |     _____________________________________________________
|   |    /
|   |   |       RX Dboard: A
|   |   |   ID: RFX900 (0x0025)
|   |   |     _____________________________________________________
|   |   |    /
|   |   |   |       RX Frontend: 0
|   |   |   |   Name: RFX900 RX
|   |   |   |   Antennas: TX/RX, RX2, CAL
|   |   |   |   Sensors: lo_locked
```



```
|   |   |   /
|   |   |   |       RX Frontend: 0
|   |   |   |   Name: RFX900 RX
|   |   |   |   Antennas: TX/RX, RX2, CAL
|   |   |   |   Sensors: lo_locked
|   |   |   |   Freq range: 750.000 to 1050.000 MHz
|   |   |   |   Gain range PGA0: 0.0 to 70.0 step 0.0 dB
|   |   |   |   Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: QI
|   |   |   |   Uses LO offset: No
|   |   |   
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       RX Codec: A
|   |   |   |   Name: ad9522
|   |   |   |   Gain range pga: 0.0 to 20.0 step 1.0 dB
|   |   
|   |    _____________________________________________________
|   |   /
|   |   |       TX DSP: 0
|   |   |   Freq range: -32.000 to 32.000 MHz
|   |   
|   |    _____________________________________________________
|   |   /
|   |   |       TX Dboard: A
|   |   |   ID: RFX900 (0x0029)
|   |   |   
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       TX Frontend: 0
|   |   |   |   Name: RFX900 TX
|   |   |   |   Antennas: TX/RX, CAL
|   |   |   |   Sensors: lo_locked
|   |   |   |   Freq range: 750.000 to 1050.000 MHz
|   |   |   |   Gain Elements: None
|   |   |   |   Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
|   |   |   |   Connection Type: IQ
|   |   |   |   Uses LO offset: Yes
|   |   |   
|   |   |    _____________________________________________________
|   |   |   /
|   |   |   |       TX Codec: A
|   |   |   |   Name: ad9522
|   |   |   |   Gain range pga: -20.0 to 0.0 step 0.1 dB
```

6.4 Starting Up the Network

Now that you have confirmed the transceiver software can communicate with the radio hardware, you can start running the OpenBTS service in the background. Do that with the following command:

```
$ sudo start openbts
```

The OpenBTS service will automatically start an instance of the transceiver software and connect to the radio hardware. Radio samples are then exchanged between the transceiver software and OpenBTS software over a local User Datagram Protocol (UDP) socket.

6.5 The Configuration System and CLI

All configuration of OpenBTS is accomplished by manipulating keys stored in a SQLite3 database. By default, this database is stored at `/etc/OpenBTS/OpenBTS.db`. Each key is defined in a schema that is compiled in OpenBTS and used to validate the values being used. One advantage afforded by this style of configuration system is that most key values can be changed and are applied to the running system within a few seconds without interrupting service. These are dynamic keys. There are also a few static keys in the configuration system that require a restart of OpenBTS to apply the change.

The easiest way to manipulate the configuration keys is via the OpenBTS command line interface (CLI). Run the following shell command to open it:

```
$ sudo /OpenBTS/OpenBTSCLI
```

You are now presented with an OpenBTS prompt. This is where commands, including configuration changes, can be executed for processing by OpenBTS. From now on, commands prefixed with ***$*** are to be executed on the Linux command line. Commands prefixed with ***OpenBTS>*** are for the OpenBTS command line. It may be convenient to have two terminal windows open so there is no need to constantly enter and exit the OpenBTS command line.

6.5.1 Changing the Band and ARFCN

The first things you must check are the radio band and Absolute Radio Frequency Channel Number (ARFCN) being used. The radio band is one of four values: 850, 900, 1800, or 1900 MHz, corresponding to the four GSM bands available around the world. An ARFCN is simply a pair of frequencies within the selected band that will be used for the transmission and reception of radio signals. Each radio band has over 100 different ARFCNs that can be used. ARFCN may also be referred to as the carrier (e.g., systems using multiple ARFCNs are multiple carrier systems). Choosing the correct band and ARFCN is important for regulatory reasons and to avoid interference with or from local carriers. You use the OpenBTS ***config*** command to inspect the current band and ARFCN settings. These configuration keys are in the `GSM.Radio` category. To view all configuration keys with the word `GSM.Radio` in their name, enter the following command:

```
OpenBTS> config GSM.Radio
GSM.Radio.ARFCNs 1     [default]
GSM.Radio.Band 900     [default]
GSM.Radio.C0 51     [default]
GSM.Radio.MaxExpectedDelaySpread 4     [default]
GSM.Radio.PowerManager.MaxAttenDB 0
GSM.Radio.PowerManager.MinAttenDB 0     [default]
GSM.Radio.RSSITarget -50     [default]
GSM.Radio.SNRTarget 10     [default]
```

The `GSM.Radio.Band` key shows that the 900 MHz band is being used and the `GSM.Radio.C0` key indicates that ARFCN #51 in that band is currently selected.

If your radio hardware does not have limitations on or optimizations for a particular frequency, you can proceed with these settings. An easy optimization for eliminating interference is to choose a band that is not used by other carriers in your country. In general, the Americas use 850 and 1900 MHz systems while the rest of the world uses 900 and 1800 MHz. Also, choose a lower frequency if possible to improve coverage with lower power. If your local regulator has assigned you a specific band and ARFCN, then you must use it. To change your GSM band, you must again use the OpenBTS config command. This time, add the desired band to the end of the command. The following example changes the band to 850 MHz:

```
OpenBTS> config GSM.Radio.Band 850
GSM.Radio.Band changed from "900" to "850"
WARNING: GSM.Radio.C0 (51) falls outside the valid range of
ARFCNs 128-251 for
GSM.Radio.Band (850)
GSM.Radio.Band is static; change takes effect on restart
```

The command confirms that the band has been changed but delivers two additional pieces of information. First, there is a warning about the ARFCN not being valid any more for the 850 MHz band. The valid range is 128–251 for 850 MHz. Second, you are informed that the ***GSM.Radio.Band*** parameter is static and cannot be applied at runtime; OpenBTS.

```
OpenBTS> config GSM.Radio.C0 166
GSM.Radio.C0 changed from "51" to "166"

GSM.Radio.C0 is static; change takes effect on restart
```

must be restarted. To fix the first warning, use the config command again to set a valid

ARFCN for the 850 MHz band:

The command confirms that the ARFCN has been changed and warns again about this parameter being static. You can now restart OpenBTS to apply the change:

```
$ sudo stop openbts

openbts stop/waiting

$ sudo start openbts

openbts start/running, process 6075
```

The service will take a few seconds to start back up and you are again free to use the OpenBTS CLI.

6.5.2 Ettus Research Radio Calibration

One main difference between the Range Networks and Ettus Research radios is in the proper value for ***GSM.Radio.RxGain***. Range Networks uses a much higher value for this parameter and if it is not adjusted, the Ettus Research equipment will not work correctly.

```
OpenBTS> devconfig GSM.Radio.RxGain 10
GSM.Radio.RxGain changed from "52" to "10"

GSM.Radio.RxGain is static; change takes effect on

restart
```

The signal being received will overdrive the demodulator. For starters, set `GSM.Radio.RxGain` to 10:

There is also a dedicated command that allows you to set this parameter without restarting OpenBTS. Use `rxgain` if you need to make fine adjustments to avoid restarting each time.

6.6 Testing Radio Frequency Environment Factors

If you've never had a project that involves RF or analog signals in general, you may be surprised by the number of things that can go wrong with them. You may actually be surprised, in the end that RF communication can work at all! RF experts in the OpenBTS community are sometimes regarded as practitioners of black magic…but we digress.

In a GSM network, two separate radio frequencies are used so the base station and handsets can communicate simultaneously in both directions. Put another way, GSM uses frequency division multiple access to establish full duplex communication. The ARFCN selected is what determines which pair of frequencies will be used. The path from the base station to the handset is known as the downlink and the path from the handset back to the base station is known as the uplink.

The next thing to look out for when setting up a new network is excess radio interference or "noise" from other sources on the uplink. If the uplink is too noisy, the signals from handsets cannot reliably be demodulated into usable information. OpenBTS will show the current level by using the `noise` command:

```
OpenBTS> noise
noise RSSI is -68 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

In this example, the detected environmental noise Received Signal Strength Indication (RSSI) is –68 dB (lower numbers are better and mean less noise is present) and the configured target RSSI level for handsets is –50 dB. This means that the base station can,
at best, receive 18 dB more energy from the handsets than the environmental noise —a very good margin meaning uplink reception issues due to noise should not be a problem.

Smaller margins between these two numbers will produce different informational messages. For example, having a margin of 10 dB or less will report:

```
WARNING: the current noise level is approaching the MS RSSI target, uplink connectivity will be extremely limited.
```

And a margin of zero or less will report:

```
WARNING: the current noise level exceeds the MS RSSI target, uplink connectivity will be impossible.
```

If either of these WARNING messages are reported, you will need to take action to reduce uplink noise and/or increase the handset transmit power.

In the future, if your handset can see the base station but can no longer connect, noise should be the first thing to check. Your configuration could still be 100% correct and functional but the radio environment may have changed, preventing communication.

6.7 Reducing Noise

If your base station radio setup does not include a frequency duplexer, the number one source of noise on the uplink can actually be the downlink signal.

Without proper duplexing to filter it out, the downlink signal is usually the closest energy source to the uplink both physically and by frequency.

6.7.1 Antenna alignment

A quick duplexer of sorts is simply aligning the antennas so they do not so readily feed into each other. If you are using rubber duck–style antennas, tilt them so they form a 90-degree angle. The radiation pattern for these antennas will then be perpendicular as shown in Figure 6-1.

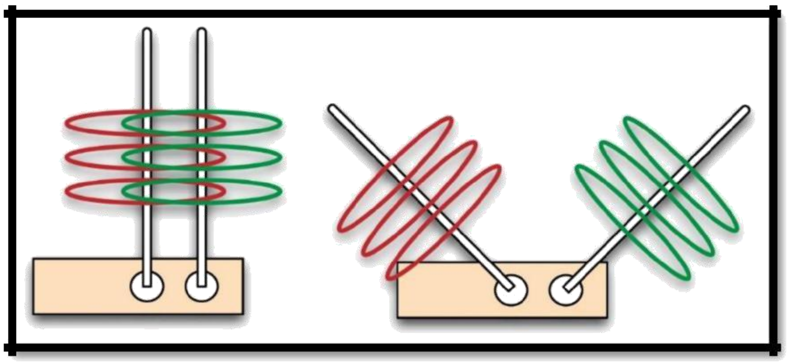


Figure 6-1 antenna alignment

If the antennas are parallel to each other, signal can efficiently flow from the transmit to the receive antenna, but when the antennas form a 90-degree angle, the signal is being transmitted on a different plane than it is being received on. Observe the change by running the ***noise*** command before and after your adjustment. This simple adjustment can reduce noise by as much as 10 dB

6.7.2 Downlink transmission power

The alignment step above reduced the flow of energy from the transmit antenna to the receive antenna. This received energy may still be too high for the uplink to be usable. Decreasing the downlink transmission power will further clean up the uplink. The coverage area lost by decreasing the downlink power is not significant in a lab environment.

Cleaner signals are preferable to strong ones. Run the ***power*** command with no arguments to see the current level. The power is reported in decibels of attenuation:

```
OpenBTS> power

Current downlink power 0 dB wrt full scale
```

To decrease the downlink transmission power, for example by 20 dB, enter the following:

```
OpenBTS> power 20

Current downlink power -20 dB wrt full scale
```

The downlink is now transmitting with 20 dB less power. Use the noise command to observe the improvement.

You can see all our OpenBTS configuration parameters in **Appendix A**. in order to make a good and clear phone call.

6.8 Steps to make a phone call for the first time configuration

In this section we are providing a full guide for any developer who wants to establish a phone call with the previously described installation by using USRP B100.

6.8.1 Searching for the Network

Now that the radio is calibrated and the settings are confirmed, you will use a handset to search for the newly created network. Each handset’s menu is different but the item is usually similar to “Carrier Selection” or “Network Selection.” The process for manually selecting a different carrier on Android is detailed in Figure 6-2.

1. Launch the “Settings” application from the Android menu system.
2. Select “More.”
3. Select “Mobile networks.”
4. Select “Network operators.” This may or may not start a search. If it does not, select “Search networks.”
5. Once the search has finished, a list of available carrier networks is presented.

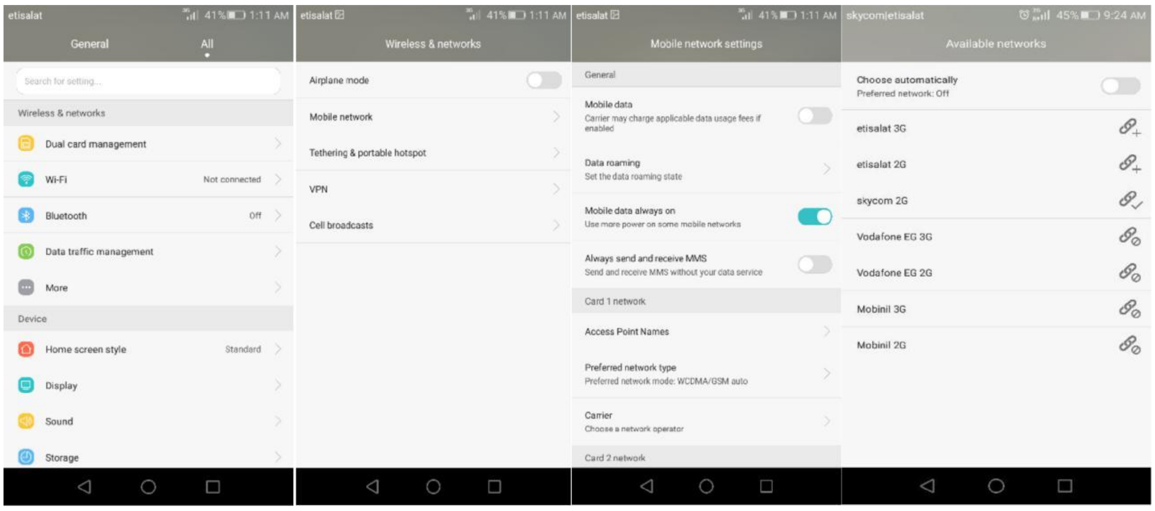


Figure 6-2 Android carrier selection

Here we see the test network in the list of selectable carriers. Depending on the handset model, firmware, and SIM used, the network ID will be displayed as "60208", "602-08" "Test PLMN 1-1", "Range" or the GSM short name of "OpenBTS." If your test network is not detected, force the search again by either reselecting the menu item, toggling airplane mode between on and off, or power cycling the handset. If that still does not work, confirm again that the handset supports the GSM band you have configured above and that the baseband is unlocked (i.e., not restricted by contract to only using a specific carrier).

6.8.2 Finding the IMSI

The main identity parameter you will be searching for is the International Mobile Subscriber Identity (IMSI). This is a 14–15-digit number stored in the SIM card and is analogous to the handset's username on the network.

Handsets will not usually divulge the IMSI of their SIM card. It can sometimes be located in a menu or through a field test mode, but this method of determining a SIM's IMSI is very cumbersome to explain. Luckily, there are other methods; OpenBTS also knows the IMSIs it has interacted with and, because you are in control of the network side, you also have access to this information.

To force an interaction between a handset and your test network, you will perform a location update request (LUR) operation on the network, analogous to a registration. This is nothing more complicated than selecting the network from the carrier selection list.

Before attempting any LURs, you need to start the SIPAuthServe daemon responsible for processing these requests:

```
$ sudo start sipauthserve

sipauthserve start/running, process 7017
```

Now, again following the steps in "Searching for the Network", bring up the carrier selection list and choose your test network. After a short time, the handset should report a registration failure.

OpenBTS remembers these LUR interactions in order to perform something called IMSI/Temporary Mobile Subscriber Identity (TMSI) exchanges. IMSI/TMSI exchanges swap the user-identifiable IMSI for a TMSI and are used to increase user privacy on the network. The exchanges are disabled by default (modify Control.LUR.SendTMSIs to enable); however, the information is still there to inspect using the ***tmsis*** command. Use it now to view all recent LUR interactions with handsets:

```
OpenBTS> tmsis

IMSI            TMSI IMEI            AUTH CREATED ACCESSED TMSI_ASSIGNED
214057715229963 -    012546629231850 0    78s     78s      0
001010000000002 -    351771054186520 1    80h     95s      0
001010000000003 -    351771053005400 1    80h     108s     0
```

Entries are sorted by time, with the top entries corresponding to the most recent inter actions. Your handset should be the top entry on this list—the most recent interaction with AUTH set to 0 because the LUR failed due to the handset not being a known subscriber. The other entries in this example are additional test handsets that have successfully performed an LUR as indicated by the AUTH column being set to 1.

6.8.3 Adding a Subscriber and OpenRegistration

6.8.3.1 *Adding a Subscriber*

You should now have all the necessary pieces of information to create a new subscriber account on your test network.

A couple of fields are still needed but are freely selectable: Name and Mobile Station International Subscriber Directory Number (MSISDN). The Name field is merely a friendly name for this subscriber so you can remember which handset or which person it is associated with. The MSISDN field is nothing more complicated than the subscriber's
phone number. Because you are not connected to the public telephone network, this can be any number you choose.

The program you need to add subscribers is ***`nmcli.py`***. It is a simple client for the Node Manager APIs and allows you to change configuration parameters, add subscribers, monitor activity, etc., all via JSON formatted commands.

`nmcli.py` is already present in your development directory—move there now to access it:

```
$ cd dev/NodeManager
```

There are two ways to add a subscriber using ***`nmcli.py`***. The first creates a subscriber that will use cached authentication:

```
$ ./nmcli.py sipauthserve subscribers create name imsi msisdn
```

The second creates a subscriber that will use full authentication:

```
$ ./nmcli.py sipauthserve subscribers create name imsi msisdn ki
```

Example:

```
$ ./nmcli.py sipauthserve subscribers create "iPhone 4" IMSI214057715229963 \
6055551234
raw request: {"command":"subscribers","action":"create","fields":
{"name":"iPhone 4","imsi":"IMSI214057715229963","msisdn":"6055551234","ki":""}}
raw response: {
        "code" : 200,
        "data" : "both ok"
}
```

Perform the same command substituting your own information to add the first subscriber to your test network.

6.8.3.2 *Open Registration*

Open Registration is an OpenBTS-specific feature that provides a Wi-Fi captive portal like implementation for mobile networks. Captive portals are familiar to anyone who has used an airport or hotel's public Wi-Fi connection. Your device can connect to the Wi-Fi network but is denied access to certain features until you answer a question, watch an advertisement, enter a pin, etc. The device is used to provision itself.

Similarly, Open Registration allows a handset to join a mobile network with initially restricted access. It may be able to dial out but the handset does not have an assigned number and as such cannot be called by other participants in the network. However, it can be used to provision its own number via SMS.

This type of network is very useful in any ad hoc installation where the users are temporary and fluid or the network itself is only temporarily needed: emergency response, remote work areas, tourist destinations, large festivals, etc. Because an administrator is not needed to create accounts and assign numbers, Open Registration networks are easier to deploy and still very useful for the users.

To enable OpenRegistration, the `Control.LUR.OpenRegistration` key must be set to a regular expression. A regular expression (sometimes written “regex”) is a way of defining a pattern to be matched.

For this book, OpenRegistration will be enabled to accept any IMSI it encounters:

```
OpenBTS> config Control.LUR.OpenRegistration  .*

Control.LUR.OpenRegistration changed from "" to ".*"
```

6.8.1 Asterisk configurations

By reaching this step we can say that all OpenBTS configuration and all our installation are done successfully but we need to know what is asterisk and why we need to configure this application in order to make a phone call.

Asterisk is a free and open source framework for building communications applications and is sponsored by Digium. Asterisk turns an ordinary computer into a communications server. Asterisk powers IP PBX systems, VoIP gateways, and conference servers and is used by small businesses, large businesses, call centers, carriers and governments worldwide.

We need to configure this application in order to route phone call between users correctly without any problems.

The first thing we need to do is to go to this directory:

```
/etc/asterisk
```

We will find some configuration files like sip.conf, extentions-range.conf ….etc.

<u>extensions-range.conf editing</u>

1- Open the text file
2- Type this line for each user:

```
exten=>phone number,1, Dial(SIP/IMSI6020300*****)
```

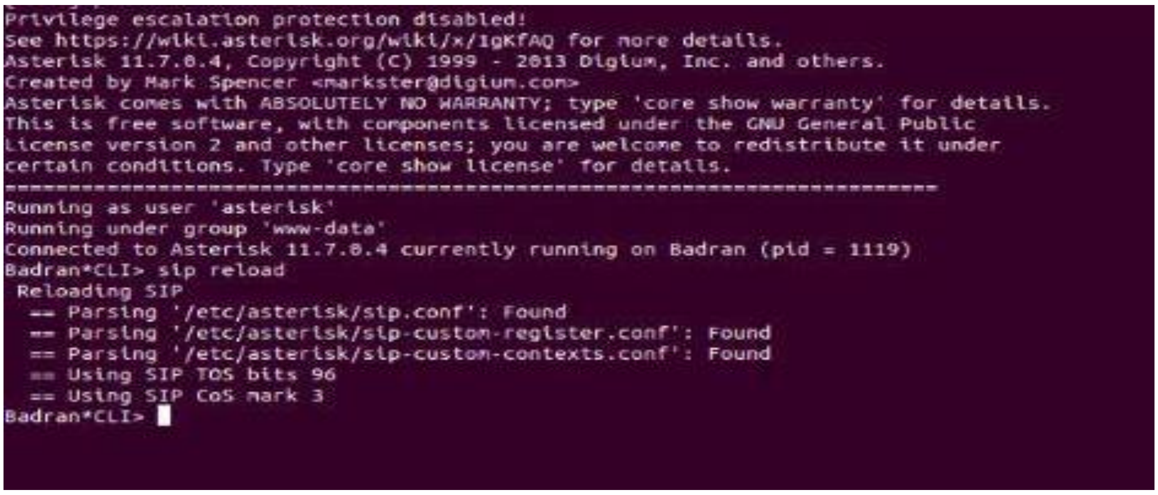
<u>sip.conf & sip-customs-contexts.conf editing</u>

1- Open the text file
2- Type this line for each user:

```
[IMSI602030*******]
callerid=01118****
canreinvite=no
host=dynamic
allow=gsm
regexten=01118***
context=phones
type=friend
```

After editing these files open asterisk and debug CLI by typing in the terminal as shown in Figure 6-3.

```
$ sudo asterisk

$ sudo asterisk –rvvvvv

> sip reload

> dialplan reload

> sip show peers
```



```
Privilege escalation protection disabled!
See https://wiki.asterisk.org/wiki/x/IgKfAQ for more details.
Asterisk 11.7.0.4, Copyright (C) 1999 - 2013 Digium, Inc. and others.
Created by Mark Spencer <markster@digium.com>
Asterisk comes with ABSOLUTELY NO WARRANTY; type 'core show warranty' for details.
This is free software, with components licensed under the GNU General Public
License version 2 and other licenses; you are welcome to redistribute it under
certain conditions. Type 'core show license' for details.
=========================================================================
Running as user 'asterisk'
Running under group 'www-data'
Connected to Asterisk 11.7.0.4 currently running on Badran (pid = 1119)
Badran*CLI> sip reload
 Reloading SIP
  == Parsing '/etc/asterisk/sip.conf': Found
  == Parsing '/etc/asterisk/sip-custom-register.conf': Found
  == Parsing '/etc/asterisk/sip-custom-contexts.conf': Found
  == Using SIP TOS bits 96
  == Using SIP CoS mark 3
Badran*CLI> 
```

Figure 6-3 : Asterisk and debug CLI

Then do the following steps:

1) search for network

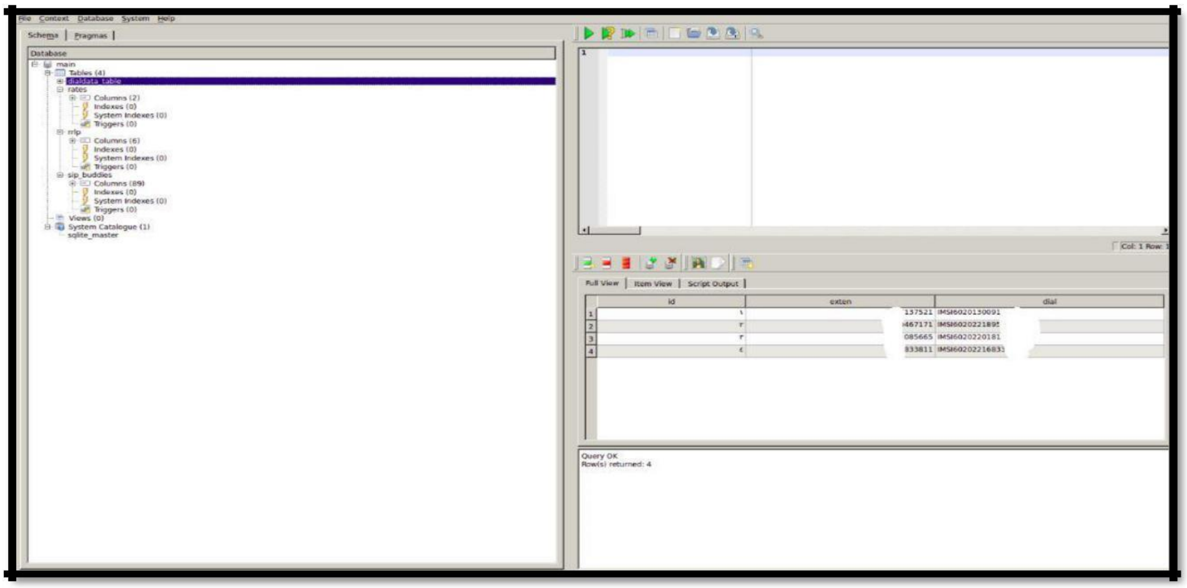
2) choose your network
3) receive SMS from 101
4) send your number to 101

After doing these steps you will be able to successfully make your first call and send your first SMS.

<u>For re-registration</u>

```
$ sudo sqliteman /var/lib/asterisk/sqlite3dir/sqlite3.db
```

Then delete all entries (dialdata , sipbuddies), After that send your number to 101 again.



	id	exten	dial
1	\	[redacted] 137521	IMSI6020130091 [redacted]
2	r	[redacted] 467171	IMSI60202218955 [redacted]
3	r	[redacted] 085665	IMSI6020220181 [redacted]
4	ε	[redacted] 833811	IMSI60202216833 [redacted]

```
Query OK
Row(s) returned: 4
```

Figure 6-4 SQlite DataBase

6.8.2 First Connection

Now when you select your test network in the connection menu, the LUR should succeed. This can be confirmed with the ***tmsis*** command in OpenBTS. The “AUTH” column will now have a “1” in the entry corresponding to your IMSI:

```
OpenBTS> tmsis
IMSI            TMSI IMEI            AUTH CREATED ACCESSED TMSI_ASSIGNED
214057715229963 -    012546629231850 1    11m     56s      0
001010000000002 -    351771054186520 1    80h     8m       0
001010000000003 -    351771053005400 1    80h     9m       0
```

Congratulations, you’ve successfully registered to your own private mobile network! Feel free to register any additional handsets you wish to use before proceeding.

6.9 Test SMS

Now that a handset has access to your network, you can perform some more interesting tests. The first is a quick test of your network’s SMS capabilities.

The component responsible for receiving, routing, and scheduling the delivery of SMS messages is SMQueue. It must be started before testing out these features; execute the following command to do so:

```
$ sudo start smqueue
smqueue start/running, process 21101
```

6.9.1 Echo SMS (411)

On your handset, compose an SMS to the number 411. This is a “shortcode” handler in SMQueue that will simply echo back whatever it receives along with some additional information about the network and subscriber account that was used.

The body of the message to 411 can be whatever you’d like, although it can be useful to use unique content for each message or sequential numbers or letters. This helps you pinpoint which message is being responded to in case an error occurs.

Once you have your message composed to 411, hit send. After a few seconds, a reply should appear (an example follows):
This indicates the following:

```
“1 queued, cell 0.1, IMSI214057715229963, phonenum 6055551234, at Sep 8 02:30:59,
Ping pong"
```

- There is one message queued for delivery.
- The base station has a load factor of 0.1.

- The message was received from IMSI 214057715229963, MSISDN 6055551234.
- The message was sent on September 8 at 02:30:59.
- The message body was “Ping pong.”

6.9.2 Direct SMS

SMS messages can also be tested directly from OpenBTS by using the ***sendsms*** command.

From the OpenBTS CLI, let’s see how it is invoked by using the ***help*** command:

```
OpenBTS> help sendsms
sendsms IMSI src# message... -- send direct SMS to IMSI on
this BTS, addressed
from source number src#.
```

Messages are sent by specifying a target IMSI, the source number the message should appear to have originated from, and the message body itself. Substitute the information for your subscriber account to compose a message and press Enter:

After a few seconds, your handset should display a new incoming message from the imaginary number 8675309 with a body of “direct SMS test.”

```
OpenBTS> sendsms 214057715229963 8675309 direct SMS test
message submitted for delivery
```

SMS messages created in this way do not route through SMQueue at all; they are sent directly out through the GSM air interface to the handset and, as such, cannot be rescheduled. If the handset is offline or unreachable, these messages are simply lost. This is why SMQueue is needed—to attempt and reschedule deliveries in the inherently unpredictable wireless environment.

6.9.3 Two-Party SMS

If you have configured more than one handset for use in your network, feel free to send a few messages back and forth between them.

6.10 Test Calls

The other service to test is voice. As with SMS, OpenBTS does not directly handle voice and requires an additional service to be run—in this case, Asterisk.

Start Asterisk now:

```
$ sudo start asterisk
asterisk start/running, process 1809
```

Using the same handset you used in the SMS tests, you will now verify a few aspects of the voice service. This is accomplished by utilizing a few test extensions that the rangeasterisk-configs package defines. An extension is an internal phone number, unreachable from the outside.

6.10.1 Test Tone Call (2602)

The first test extension you will use plays back a constant tone. This might not sound too exciting but does confirm many things about the network:

- Asterisk is running and reachable.
- Call routing is working as expected.
- Downlink audio is functional.

Call 2602 with your handset now.

As you listen to the tone, listen for changes in pitch. These changes in pitch are due to missing information in the downlink voice stream path, similar to packet loss. In the field, this is the primary use for the test tone extension: testing downlink quality. A downlink loss of 3% is normal in production networks, with losses of 5%–7% still pro- viding an understandable conversation.

6.10.2 Echo Call (2600)

The next test extension creates an “echo call.” Basically, all audio that Asterisk receives will be immediately echoed back to the sender. In addition to confirming the items listed for the test tone call, the echo call will reveal any delay or uplink quality issues present in your network.

Call 2600 with your handset now. As you speak into the microphone, you should hear yourself very shortly afterward in the earpiece. A little delay is normal, but longer delays lead to an experience more like using a walkie-talkie. The human brain can deal with delays up to about 200 ms without trouble. Beyond that, the conversation starts to break down and both sides stop speaking because it becomes uncomfortable.

6.10.3 Two-Party Call

If you have configured more than one handset for use in your network, feel free to place some calls between them. Verify that the source numbers are correct when receiving a call.

Chapter 7: Problems and solutions

1. **problem**: there was problem with Ubuntu 14.04 (64-bit) as it needs updated version of openbts (version 5 or more) while version 5 isn't open source.

- ✓ **Solution**: using different version of Ubuntu (Ubuntu 12)

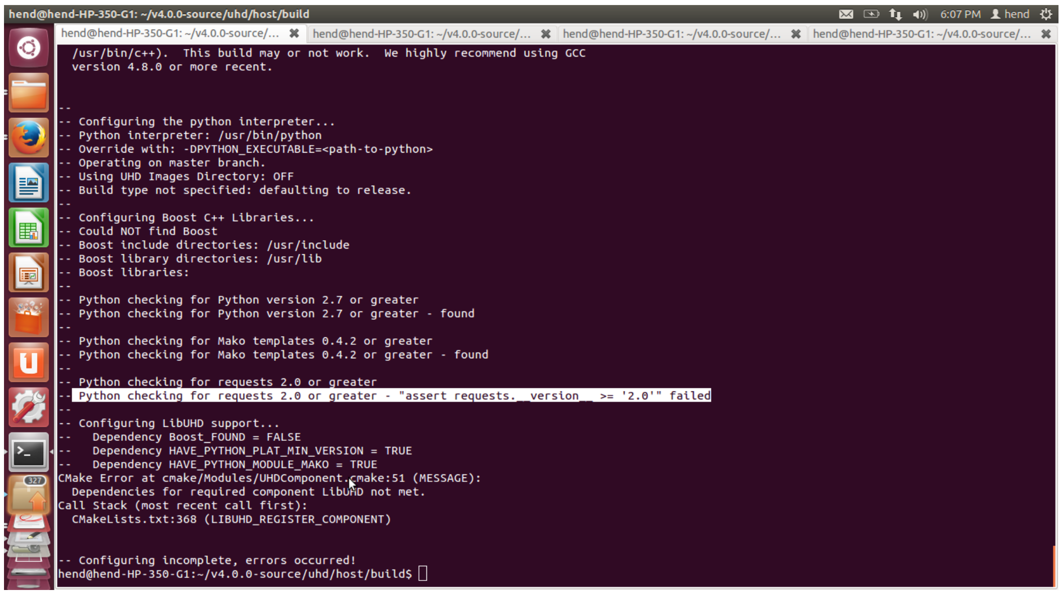
2. **problem**: using Ubuntu 12 (64-bit) all the applications and SMS run successfully but the problem with asterisk (version 1.8) so call didn't run as the asterisk was old version and 32-bit while compiler was 64-bit so all the trials to upgrade the version of asterisk fail.

- ✓ **Solution**: using Ubuntu 12 but 32-bit in which it was successful to use asterisk 11.25 so the call runs successfully.

3. **problem**: while installing libraries there were a lot of dependent libraries which needs to be installed which take a long time using (sudo apt-get install (name of package/library) as this command doesn't install each library with its dependent libraries.

- ✓ **Solution**: downloading libraries using the following command (sudo dpkg -I(name of the library)) which downloads each library with its dependent libraries or using this command (sudo apt-get -y install git swig cmake doxygen build-essential libboost-all-dev libtool libusb-1.0-0 libusb-1.0-0-dev libudev-dev libncurses5-dev libfftw3-bin libfftw3-dev libfftw3-doc libcppunit-1.13-0 libcppunit-dev libcppunit-doc ncurses-bin cpufrequtils python-numpy python-numpy-doc python-numpy-dbg python-scipy python-docutils qt4-bin-dbg qt4-default qt4-doc libqt4-dev libqt4-dev-bin python-qt4 python-qt4-dbg python-qt4-dev python-qt4-doc python-qt4-doc libfftw3-bin libfftw3-dev libfftw3-doc ncurses-bin libncurses5 libncurses5-dev libncurses5-dbg libfontconfig1-dev libxrender-dev libpulse-dev swig g++ automake autoconf libtool python-dev libfftw3-dev libcppunit-dev libboost-all-dev libusb-dev libusb-1.0-0-dev fort77 libsdl1.2-dev python-wxgtk2.8 git-core libqt4-dev python-numpy ccache python-opengl libgsl0-dev python-cheetah python-mako python-lxml doxygen qt4-default qt4-dev-tools libusb-1.0-0-dev libqwt5-qt4-dev libqwtplot3d-qt4-dev pyqt4-dev-tools python-qwt5-qt4 cmake git-core wget libxi-dev gtk2-engines-pixbuf r-base-dev python-tk liborc-0.4-0 liborc-0.4-dev libasound2-dev python-gtk2 libzmq1 libzmq-dev python-requests python-sphinx)which downloads most of linux libraries but the problem that this commands downloads alot of libraries which wasn’t needed and take alot of memory.

4. **problem**: UHD the commands used to install latest version while the operating system and applications are old versions.The following figure 7-1 shows the error appeared.

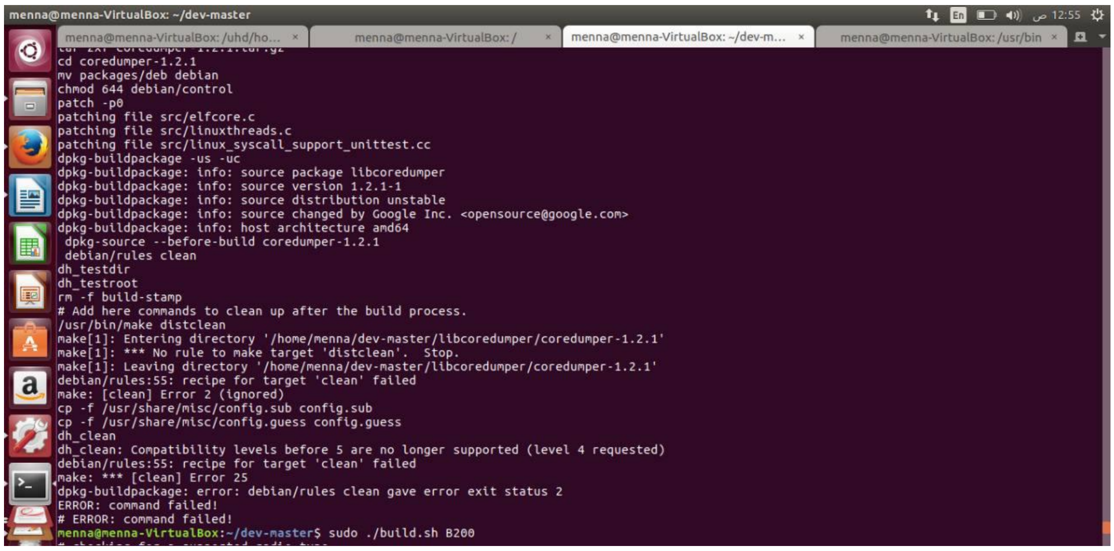

hend@hend-HP-350-G1: ~/v4.0.0-source/uhd/host/build
6:07 PM
hend
hend@hend-HP-350-G1: ~/v4.0.0-source/...
hend@hend-HP-350-G1: ~/v4.0.0-source/...
hend@hend-HP-350-G1: ~/v4.0.0-source/...
hend@hend-HP-350-G1: ~/v4.0.0-source/...

```
  /usr/bin/c++).  This build may or not work.  We highly recommend using GCC
  version 4.8.0 or more recent.


--
-- Configuring the python interpreter...
-- Python interpreter: /usr/bin/python
-- Override with: -DPYTHON_EXECUTABLE=<path-to-python>
-- Operating on master branch.
-- Using UHD Images Directory: OFF
-- Build type not specified: defaulting to release.
--
-- Configuring Boost C++ Libraries...
-- Could NOT find Boost
-- Boost include directories: /usr/include
-- Boost library directories: /usr/lib
-- Boost libraries:
--
-- Python checking for Python version 2.7 or greater
-- Python checking for Python version 2.7 or greater - found
--
-- Python checking for Mako templates 0.4.2 or greater
-- Python checking for Mako templates 0.4.2 or greater - found
--
-- Python checking for requests 2.0 or greater
-- Python checking for requests 2.0 or greater - "assert requests.__version__ >= '2.0'" failed
--
-- Configuring LibUHD support...
--   Dependency Boost_FOUND = FALSE
--   Dependency HAVE_PYTHON_PLAT_MIN_VERSION = TRUE
--   Dependency HAVE_PYTHON_MODULE_MAKO = TRUE
CMake Error at cmake/Modules/UHDComponent.cmake:51 (MESSAGE):
  Dependencies for required component LibUHD not met.
Call Stack (most recent call first):
  CMakeLists.txt:368 (LIBUHD_REGISTER_COMPONENT)


-- Configuring incomplete, errors occurred!
hend@hend-HP-350-G1:~/v4.0.0-source/uhd/host/build$ 
```

Figure 7-1 error UHD

- ✓ **Solution**: download old version of UHD (version 003.009) manually as has been mentioned in chapter 5 of installations.

5. **problem**: The problem with USRP B200 as the project runs using B100 with Ubuntu 12.04, when we asked the support of Ettus research about the problem they told us that the problem that the USRP is a new version so we must use (Ubuntu 16.04- code openbts 5 -UHD 003.009.11), when we tried this the error shown in the following Figure 7-2 error appeared.



```
cd coredumper-1.2.1
mv packages/deb debian
chmod 644 debian/control
patch -p0
patching file src/elfcore.c
patching file src/linuxthreads.c
patching file src/linux_syscall_support_unittest.cc
dpkg-buildpackage -us -uc
dpkg-buildpackage: info: source package libcoredumper
dpkg-buildpackage: info: source version 1.2.1-1
dpkg-buildpackage: info: source distribution unstable
dpkg-buildpackage: info: source changed by Google Inc. <opensource@google.com>
dpkg-buildpackage: info: host architecture amd64
 dpkg-source --before-build coredumper-1.2.1
 debian/rules clean
dh_testdir
dh_testroot
rm -f build-stamp
# Add here commands to clean up after the build process.
/usr/bin/make distclean
make[1]: Entering directory '/home/menna/dev-master/libcoredumper/coredumper-1.2.1'
make[1]: *** No rule to make target 'distclean'.  Stop.
make[1]: Leaving directory '/home/menna/dev-master/libcoredumper/coredumper-1.2.1'
debian/rules:55: recipe for target 'clean' failed
make: [clean] Error 2 (ignored)
cp -f /usr/share/misc/config.sub config.sub
cp -f /usr/share/misc/config.guess config.guess
dh_clean
dh_clean: Compatibility levels before 5 are no longer supported (level 4 requested)
debian/rules:55: recipe for target 'clean' failed
make: *** [clean] Error 25
dpkg-buildpackage: error: debian/rules clean gave error exit status 2
ERROR: command failed!
# ERROR: command failed!
menna@menna-VirtualBox:~/dev-master$ sudo ./build.sh B200
```

Figure 7-2 error in Ubuntu 16.04

✓ **Solution**: using Ubuntu 14 .04

Chapter 8: GSM Gateway

8.1 Motivation

we are discussing all the ways to go outside the GSM network in order to communicate with outside world by using GSM gateway.

8.1.1 What is WGW1002?

OpenVox VoxStack Series GSM Gateway is an open source asterisk-based GSM VoIP Gateway solution for SMBs and SOHOs. With friendly GUI and unique modular design, users may easily setup their customized Gateway. Also secondary development can be completed through AMI (Asterisk Management Interface).

There are three models with VoxStack series GSM Gateway WGW1002, VS-GW1202-8G and VS-GW1600-20G. There are 2 channels in WGW1002 and 4/8 GSM channels in VS-GW1202-8G. The Modular Design GSM Gateways are ranging from 4 up to 20 GSM channels on the VS-GW1600 series gateways, developed for interconnecting the GSM cellular networks with a wide selection of codecs and signaling protocol, including G.711A, G.711U, G.729, G.722, G.723, G.726 and GSM to quickly reduce communication expenses and maximize cost-savings. With the unique design of the VoxStack gateway, it can support hot-swap for both SIM cards and GSM gateway modules. Users can simply add or remove the modules for hardware expansion or exchange.

The VoxStack gateway designs with 2 LAN switch boards to provide stack ability on the hardware upgrade, and five VS-GWM400G modules which are independent with each other, so each one has a GUI configuration web. If you connect to ETH1, you can access Board 1 only and access other boards with different port numbers which can avoid IP conflict. Otherwise if you connect to ETH2, you can access different Boards with different IP addresses.

GSM Gateway WGW1002 support SMS messages sending, receiving, group sending and SMS to E-mail. The GSM gateway will be 100% compatible with Asterisk, Elastix, trixbox, 3CX, FreeSWITCH SIP server and VOS VoIP operating platform.

Sample Application

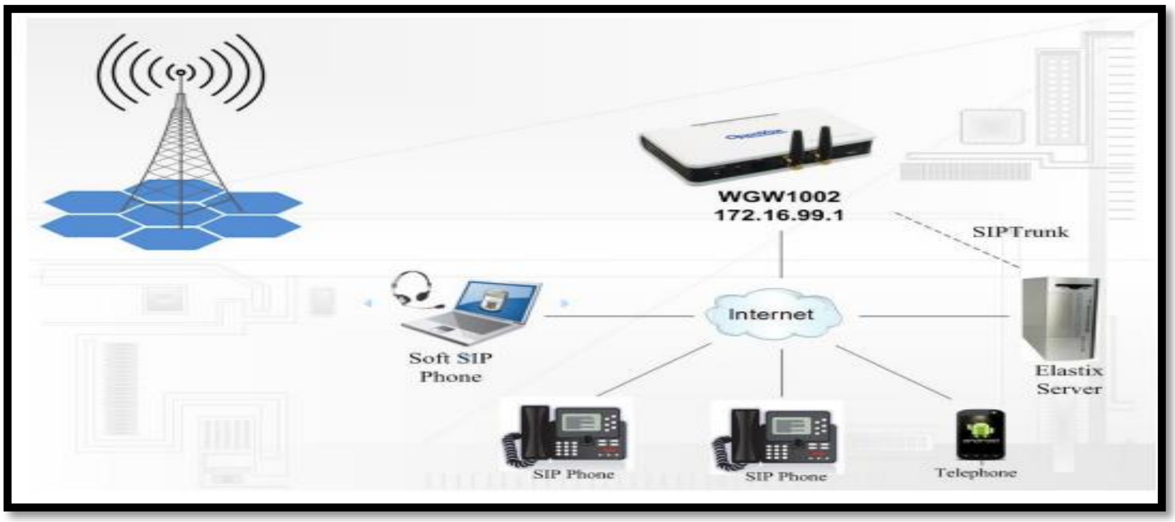


Figure 8-1 Topological Graph

8.1.2 Product Appearance

The Figure 8-2 below is appearance of VS-GW1202-8G.

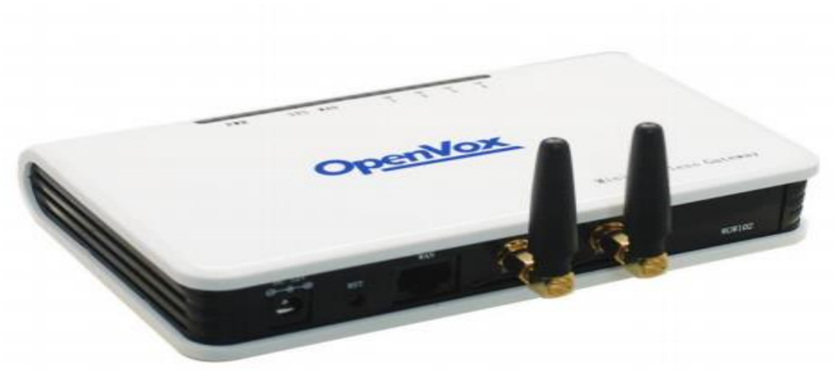


Figure 8-2 Product Appearance

8.1.2.1 Main Features

- Based on Asterisk
- Editable Asterisk configuration file

- Wide selection of codecs and signaling protocol
- Support SMS sending, receiving, group sending
- Support transferring SMS to E-mail
- Support SMS automatically resent
- Support SMS remotely controlling gateway
- Support USSD service
- Support IMEI modification
- Support PIN identification
- Support unlimited routing rules and flexible routing settings
- Hot-swap
- Stable performance, flexible dialing, friendly GUI

8.1.2.2 Physical Information

Table 8-1 Description of Physical Information

Weight	16cm*10.1cm*3.1cm
Size	237g
Frequency	GSM 850/900/1800/1900MHz
Temperature	-20~70°C (Storage)
	0~40°C (Operation)
Operation humidity	10%~90% non-condensing
Power source	12V DC/2.33A
Max power	6W
LAN port	1

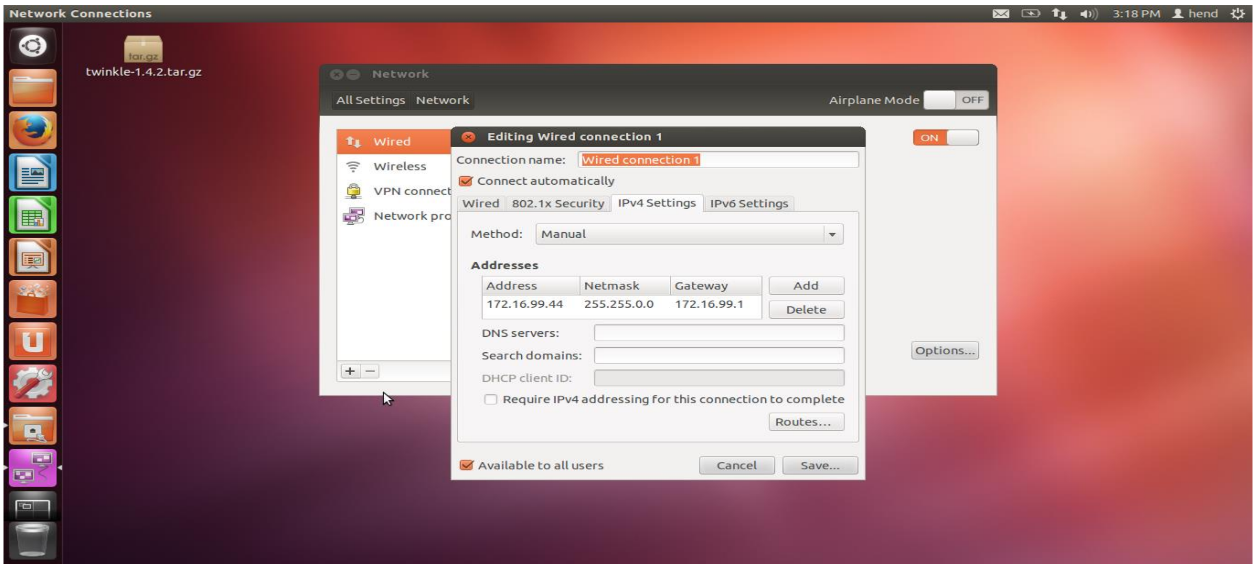
8.1.3 Software Default

- ❖ IP: 172.16.99.1
- ❖ Username: admin
- ❖ Password: admin

Firstly, to can log in as shown in figure 8-3 to configure the module Go to setting =>Network=>wired connection, enable it and Go to pv4 setting: -

Enter Gateway Address 172.16.99.1 & net mask 255.255.0.0 & give an IP Address in the range of The Gateway IP, let’s make it 172.16.99.44

Note save this IP Address in your head, we will need it soon.



Connection name: Wired connection 1

☑ Connect automatically

Method: Manual

Addresses

Address	Netmask	Gateway
172.16.99.44	255.255.0.0	172.16.99.1

DNS servers:

Search domains:

DHCP client ID:

☐ Require IPv4 addressing for this connection to complete

☑ Available to all users

Figure 8-3 how to log in

Now, Please enter the default IP "172.16.99.1" in your browser to scan and configure the module.

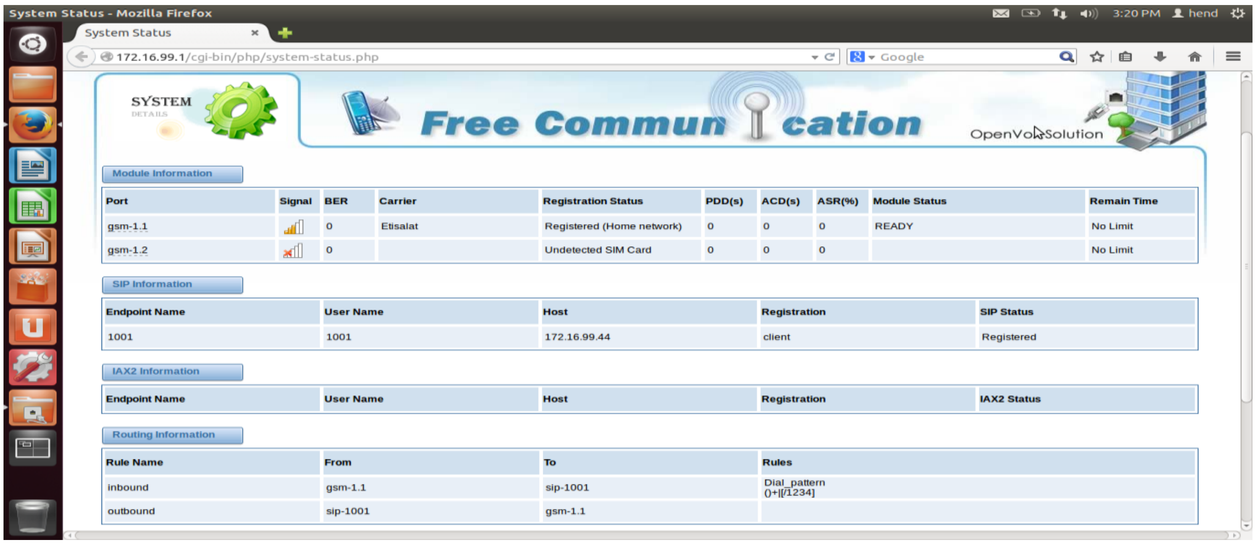
Log in:

- ❖ IP: 172.16.99.1
- ❖ Username: admin
- ❖ Password: admin

8.2 System Status

On the “Status” page as shown in figure 8-4, you will find all GSM, SIP, Routing, Network information and status.

Any change you make, you will see it in this page, in the end of your configuration, you can check I your status page like this.



Module Information

Port	Signal	BER	Carrier	Registration Status	PDD(s)	ACD(s)	ASR(%)	Module Status	Remain Time
gsm-1.1		0	Etisalat	Registered (Home network)	0	0	0	READY	No Limit
gsm-1.2		0		Undetected SIM Card	0	0	0		No Limit

SIP Information

Endpoint Name	User Name	Host	Registration	SIP Status
1001	1001	172.16.99.44	client	Registered

IAX2 Information

Endpoint Name	User Name	Host	Registration	IAX2 Status

Routing Information

Rule Name	From	To	Rules
inbound	gsm-1.1	sip-1001	Dial_pattern ()+|[/1234]
outbound	sip-1001	gsm-1.1	

Figure 8-4 System Status

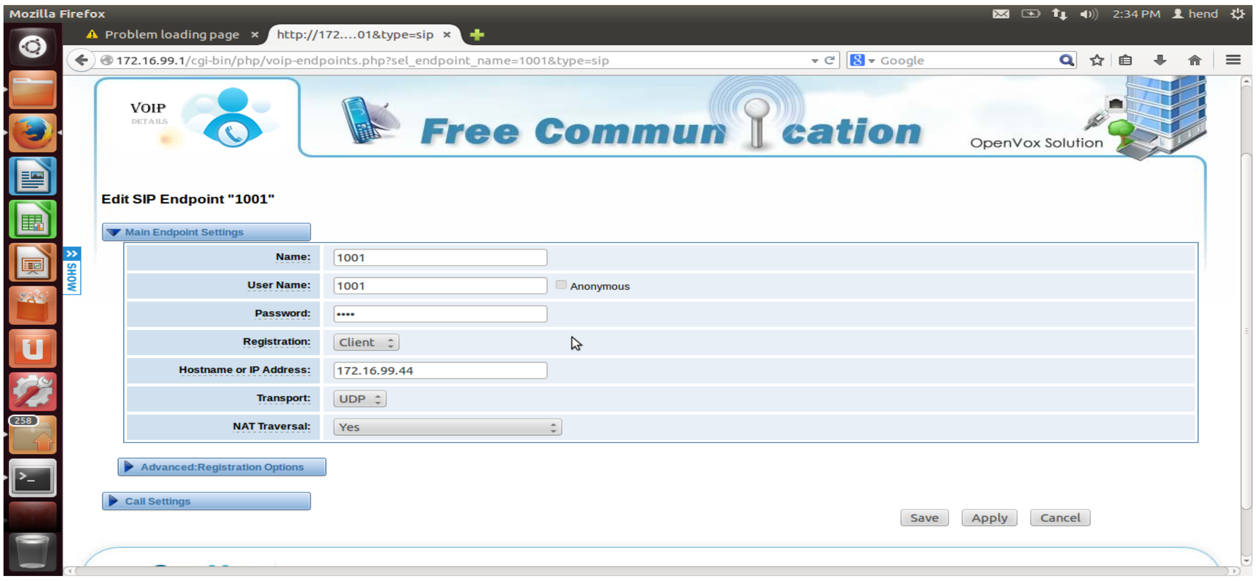
Now, I will talk about the most important configuration you must do . But there ara a lot of configurations you can do it if you need. these configurations you can find it here.

$http://www.openvox.cn/products/voip-gateways/176/64/3g-gateway/wgw1002g-detail.html#download

8.3 Voip & SIP Endpoints

As Shown in figure 8-5 everything about your SIP, you can see status of each SIP.

You can click Add New SIP Endpoint k button to add a new SIP endpoint, and if you want to modify existed endpoints, you can click button. Don't forget to click save and apply after any change.


Edit SIP Endpoint "1001"

Main Endpoint Settings

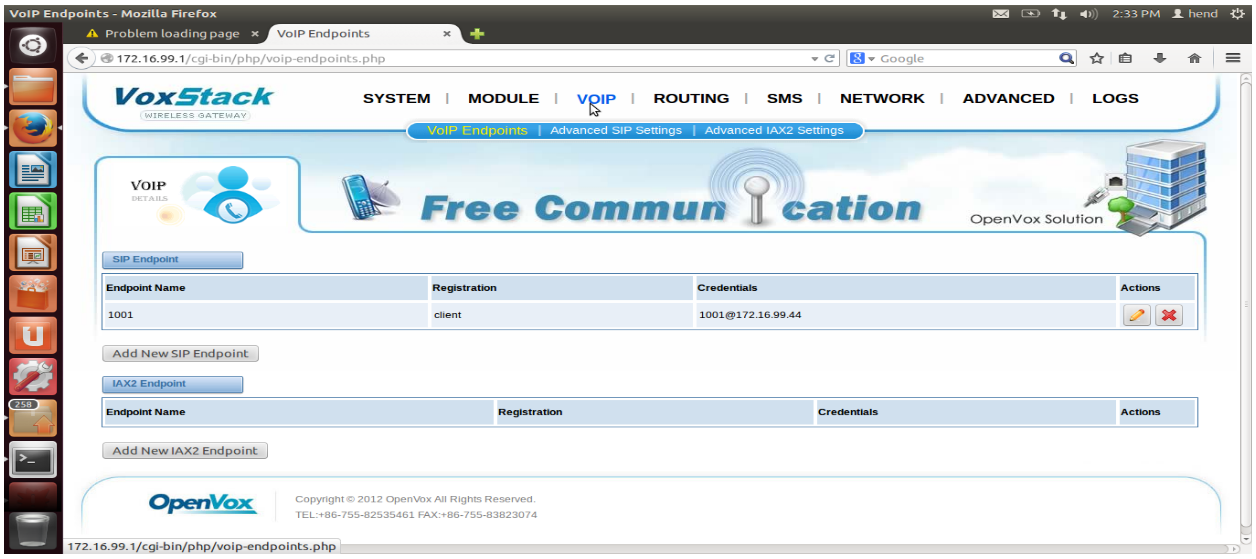
Name:	1001
User Name:	1001 ☐ Anonymous
Password:	••••
Registration:	Client
Hostname or IP Address:	172.16.99.44
Transport:	UDP
NAT Traversal:	Yes

Advanced:Registration Options

Call Settings

Save Apply Cancel

Figure 8-5add sip end point


SYSTEM | MODULE | VOIP | ROUTING | SMS | NETWORK | ADVANCED | LOGS

VoIP Endpoints | Advanced SIP Settings | Advanced IAX2 Settings

SIP Endpoint

Endpoint Name	Registration	Credentials	Actions
1001	client	1001@172.16.99.44	

Add New SIP Endpoint

IAX2 Endpoint

Endpoint Name	Registration	Credentials	Actions

Add New IAX2 Endpoint

Copyright © 2012 OpenVox All Rights Reserved.
TEL:+86-755-82535461 FAX:+86-755-83823074

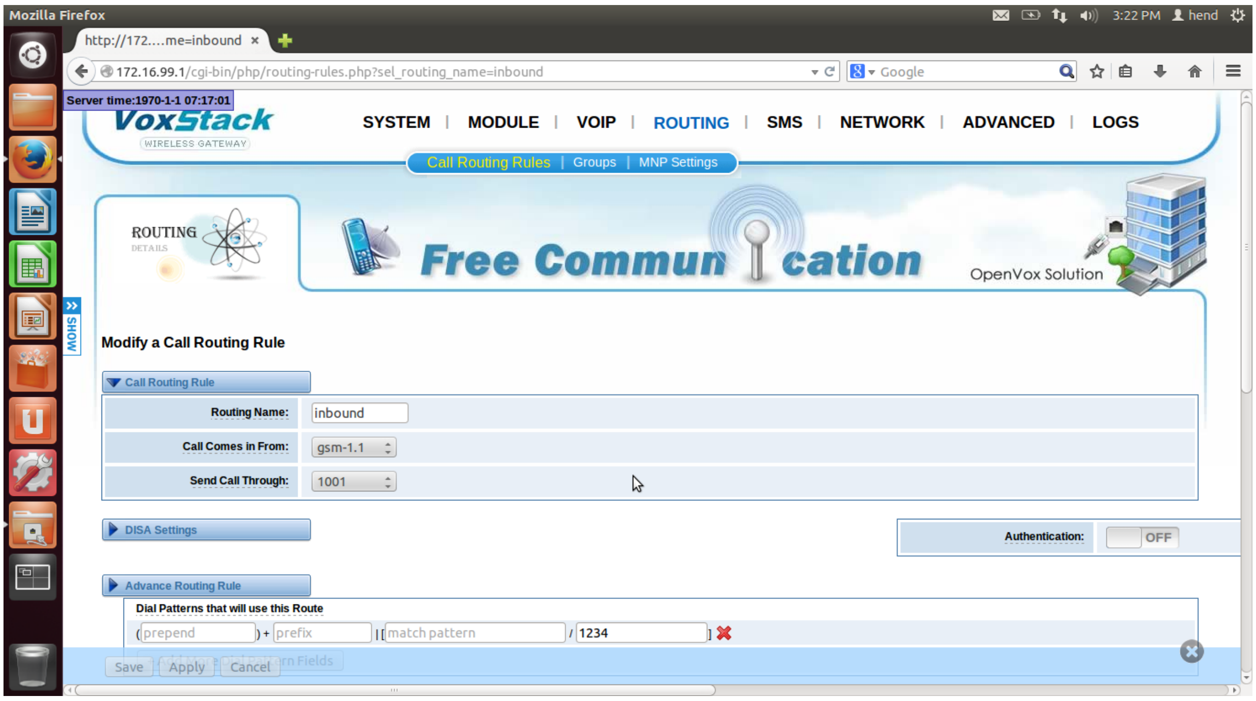
Figure 8-6 voip endpoint

8.4 Call routing rules

You are allowed to set up new routing rule by New Call Routing Rule , and after setting routing rules, move rules' order by pulling up and down, click button to edit the routing and to delete it. Finally click the Save Orders button to save what you set. Rules shows current routing rules. Otherwise you can set up unlimited routing rules.

Table 8-2 Definition of Routing Options

Options	Definition
Routing Name	The name of this route. Should be used to describe what types of calls this route matches (for example, 'SIP2GSM' or 'GSM2SIP').
Call Comes in From	The launching point of incoming calls.
Send Call Through	The destination to receive the incoming calls.


Mozilla Firefox
http://172.16.99.1/cgi-bin/php/routing-rules.php?sel_routing_name=inbound
Server time:1970-1-1 07:17:01
SYSTEM | MODULE | VOIP | ROUTING | SMS | NETWORK | ADVANCED | LOGS
Call Routing Rules | Groups | MNP Settings

Modify a Call Routing Rule

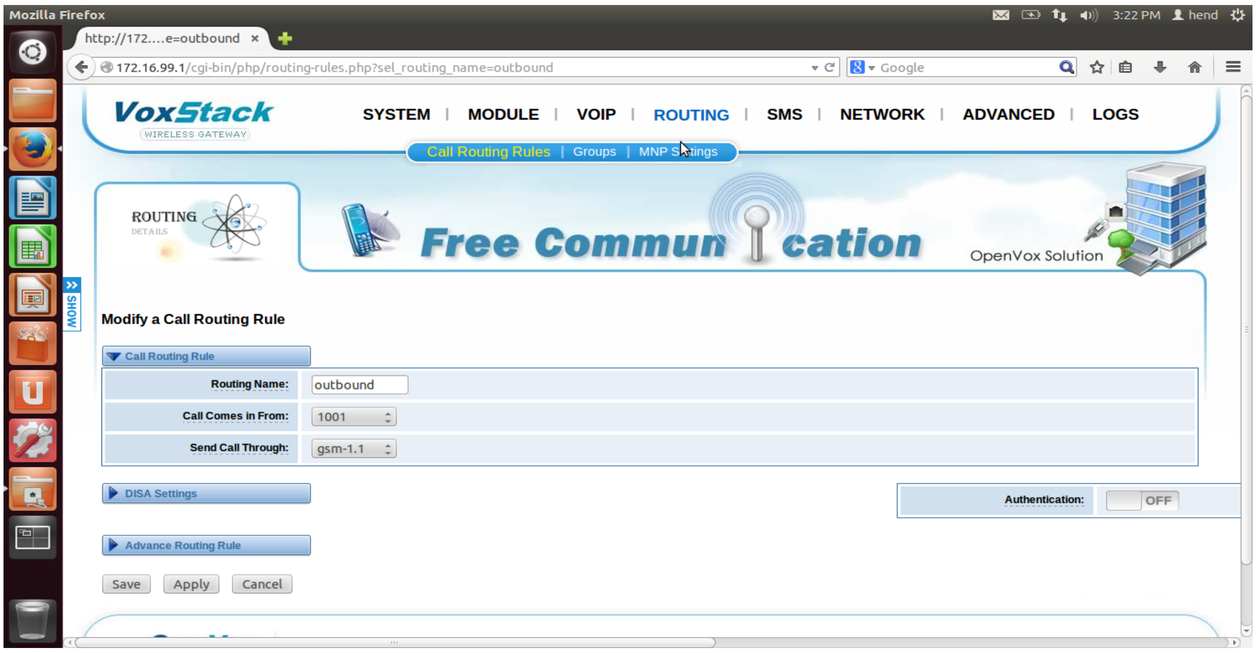
Call Routing Rule
Routing Name: inbound
Call Comes in From: gsm-1.1
Send Call Through: 1001

DISA Settings
Authentication: OFF

Advance Routing Rule
Dial Patterns that will use this Route
(prepend) + prefix | [match pattern / 1234]

Save Apply Cancel

Figure 8-7set up Routing Rule(inbound)


SYSTEM | MODULE | VOIP | ROUTING | SMS | NETWORK | ADVANCED | LOGS
Call Routing Rules | Groups | MNP Settings

Modify a Call Routing Rule

Call Routing Rule

Routing Name:	outbound
Call Comes in From:	1001
Send Call Through:	gsm-1.1

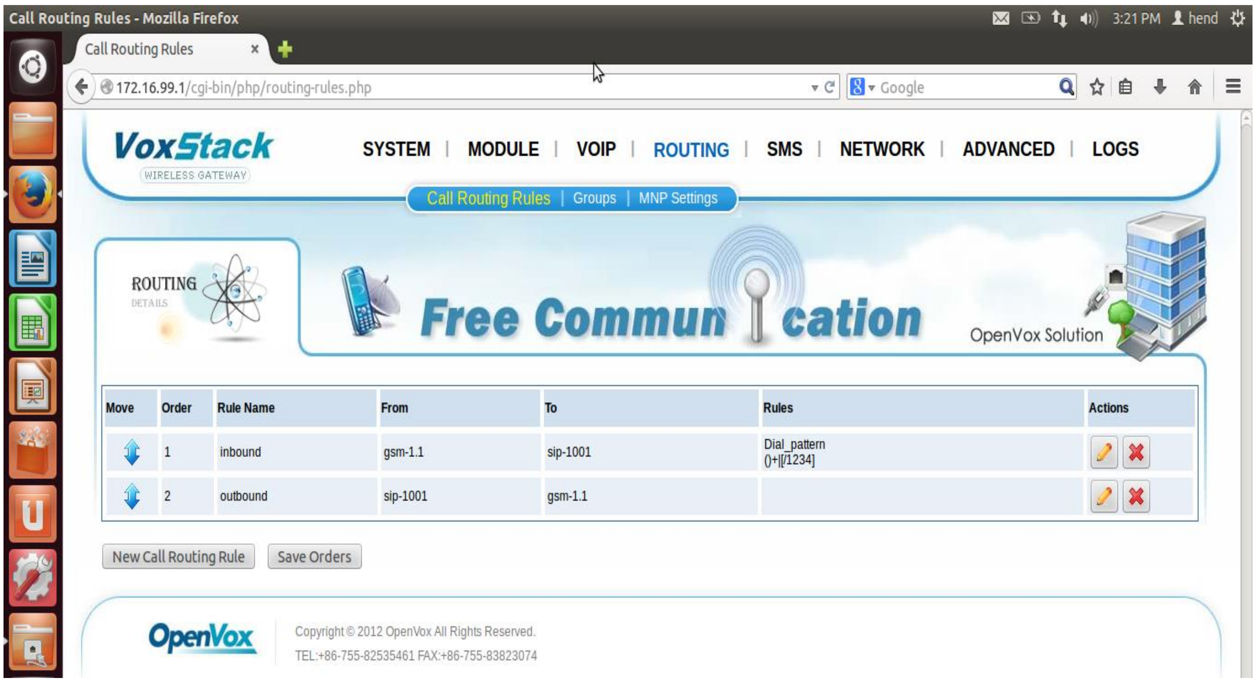
DISA Settings

Authentication: OFF

Advance Routing Rule

Figure 8-8 set up Routing Rule (outbound)

The figure 8-8 above realizes that calls from “support” SIP endpoints which you have registered will be transferred to gsm-1. When “Call Comes in from” and vice versa.


SYSTEM | MODULE | VOIP | ROUTING | SMS | NETWORK | ADVANCED | LOGS
Call Routing Rules | Groups | MNP Settings

Move	Order	Rule Name	From	To	Rules	Actions
	1	inbound	gsm-1.1	sip-1001	Dial_pattern ()+|[/1234]	
	2	outbound	sip-1001	gsm-1.1		

New Call Routing Rule
Save Orders

Copyright © 2012 OpenVox All Rights Reserved.
TEL:+86-755-82535461 FAX:+86-755-83823074

Figure 8-9 Routing Rules

8.5 Softphone

To achieve our purpose of the project, actually we don’t need softphone but we use it as an easier way to enable us to make phone calls, after that we connect gateway with USRB without using a softphone. So you can skip this section if you need.

To make sure that Gateway works efficiently, we use Twinkle as a softphone to make calls and send sms, then we made configurations to connect the getaway with.

A softphone is a piece of software that simulates the action of a telephone and allows you to make, to receive and to manage voice calls over the Internet. Softphones normally run on computers, tablets, PC, and smartphones, and are necessary for placing VoIP calls and video calls.

8.5.1 Parts of a Softphone

A softphone has the following parts:

- An interface, which acts as a platform for communication between the user and the computer or device. It offers the user a dial pad to dial numbers, and in some cases, a keyboard or keyboard to enter names for new contacts and searches. The interface also includes control buttons to manage calls and presence. It works with audio input and output of the device in order to play sounds and capture audio during calls.
- An engine for processing calls, with modules in a communication API that allow calls to be placed and received in a particular protocol.
- A set of codecs that allow voice data to be encoded between analog (as we hear it and as we speak naturally) and digital formats. Codecs also compress the data so that they can easily be transferred over the Internet.
- A contact list, which is a list of numbers and names that make it easy for a user to keep track of numbers and management correspondents.

8.5.2 Types of Softphone

Softphones have evolved over the years throughout the development of the VoIP industry. In the early days of VoIP, softphone were replications of the traditional phone over a screen. Nowadays, they are incorporated as the basic interface for communication apps.

Softphones differ based on their functionalities, on the purpose of their use, on the sophistication and complexity of the protocol they are under, and on the features offered.

For instance, a softphone that is designed for business purposes is likely to have a bulky interface and a lot of features with rich menus and options.

On the other hands, smartphone and chat apps have very simple and basic softphone interfaces that require only one or two touches of the finger to initiate a call.

8.5.3 Communication protocols

To communicate, both end-points must support the same Voice-over-IP protocol, and at least one common audio codec.

Many service providers use the Session Initiation Protocol (SIP) standardized by the Internet Engineering Task Force (IETF). Skype, a popular service, uses proprietary protocols, and Google Talk leverages the Extensible Messaging and Presence Protocol (XMPP).

Some softphones also support the Inter-Asterisk eXchange protocol (IAX), a protocol supported by the open-source software application Asterisk.

8.5.4 Features

A typical softphone has all standard telephony features (DND, Mute, DTMF, Flash, Hold, Transfer etc.) and often additional features typical for online messaging, such as user presence indication, video, wide-band audio. Softphones provide a variety of audio codecs, a typical minimum set is G.711 and G.729.

8.5.5 Requirements

To make voice calls via the Internet, a user typically requires the following:

- A modern PC with a microphone and speaker, or with a headset, or with USBphone.
- Reliable high-speed Internet connectivity like Digital subscriber line (DSL), or cable service.
- Account with an Internet telephony service provider or IP PBX provider.
- Mobile or landline phone

8.5.6 Examples of Softphones

A good example of a business softphone is Counterpath's X-Lite which is free but full of features. A more enhanced version is the paid Bria.

Besides, Skype has a softphone incorporated in its interface. Given that Skype users are identified through their user names and not numbers, the dial pad is not often used. But for Skype Out calls, where users will have to dial the numbers of landlines and mobile devices they are contacting, a very basic interface is used. Likewise, all other VoIP apps.

More sophisticated softphones do not resemble much of a phone, in that they use other methods of selecting contacts and dialing. For instance, some softphones allow users to say the names of the contacts and through voice recognition place the calls. In this project we use Twinkle as softphone to make calls and send SMS.

8.5.7 Twinkle

Twinkle is a softphone for your voice over IP and a free and open-source app for voice communications over Voice over IP (VoIP) protocol. It is designed for Linux operating systems and uses the Qt toolkit for its graphical user interface. For call signaling it employs the Session Initiation Protocol (SIP). It also features direct IP-to-IP calls. Media streams are transmitted via the Real-time Transport Protocol (RTP) which may be encrypted with the Secure Real-time Transport Protocol (SRTP) and the ZRTP security protocols.

8.5.7.1 Technical Features

In addition to making basic voice calls Twinkle provides you the following features:

- 2 call appearances (lines)
- Multiple active call identities

- Custom ring tones
- Call Waiting
- Call Hold
- 3-way conference calling
- Mute
- Call redirection on demand
- Call redirection unconditional
- Call redirection when busy
- Call redirection no answer
- Reject call redirection request
- Blind call transfer
- Call transfer with consultation (attended call transfer)
- Reject call transfer request
- Call reject
- Repeat last call
- Do not disturb
- Auto answer
- Message Waiting Indication
- Voice mail speed dial
- User definable scripts triggered on call events
- E.g. to implement selective call reject or distinctive ringing
- RFC 2833 DTMF events
- In band DTMF
- Out-of-band DTMF (SIP INFO)
- STUN support for NAT traversal
- Send NAT keep alive packets when using STUN
- NAT traversal through static provisioning
- Persistent TCP connections for NAT traversal
- Missed call indication

- History of call detail records for incoming, outgoing, successful and missed calls
- DNS SRV support
- Automatic failover to an alternate server if a server is unavailable
- Other programs can originate a SIP call via Twinkle, e.g. call from address book
- System tray icon
- System tray menu to quickly originate and answer calls while Twinkle stays hidden
- User definable number conversion rules
- Simple address book
- Support for UDP and TCP as transport for SIP
- Presence
- Instant messaging
- Simple file transfer with instant message
- Instant message composition indication
- Command line interface (CLI)

8.5.7.2 VoIP security

- Secure voice communication by ZRTP/SRTP
- MD5 digest authentication support for all SIP requests
- AKAv1-MD5 digest authentication support for all SIP requests
- Identity hiding

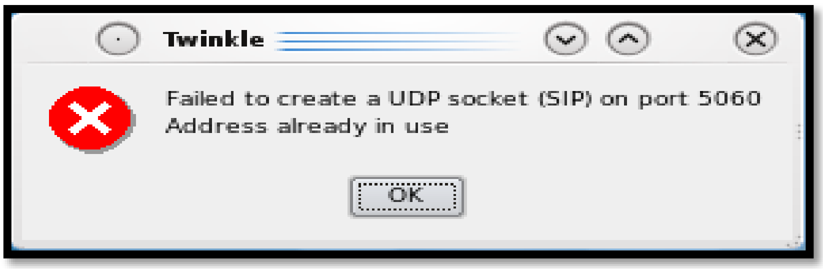
8.5.7.3 Supported audio formats

- G.711 A-law: 64 kbit/s payload, 8 kHz sampling rate
- G.711 μ-law: 64 kbit/s payload, 8 kHz sampling rate

- G.726: 16, 24, 32 or 40 kbit/s payload, 8 kHz sampling rate
- GSM: 13 kbit/s payload, 8 kHz sampling rate
- G.729: 8 kbit/s payload, 8 kHz sampling rate
- iLBC: 13.3 or 15.2 kbit/s payload, 8 kHz sampling rate
- Speex narrow band: 15.2 kbit/s payload, 8 kHz sampling rate
- Speex wide band: 28 kbit/s payload, 16 kHz sampling rate
- Speex ultra wide band: 36 kbit/s payload, 32 kHz sampling rate

8.5.7.4 installation

First when we install twinkle we follow the steps in Twinkle's official site but an error massage appeared as shown in figure 8-10

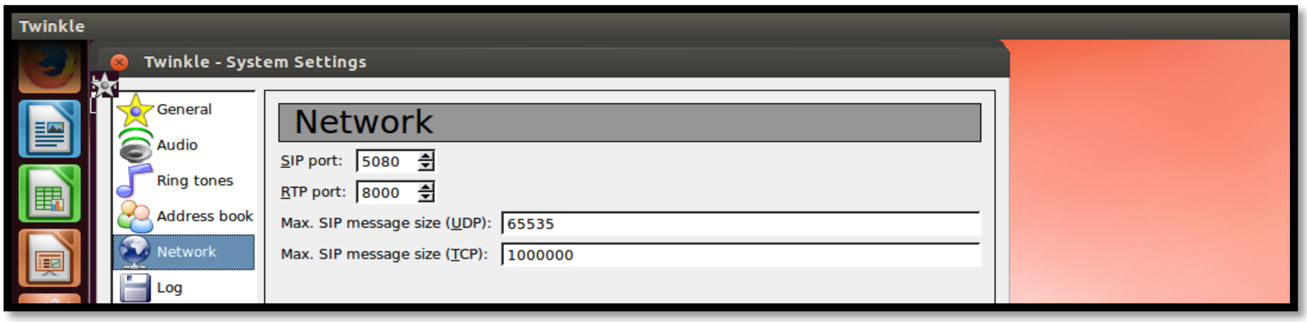


Twinkle

```
Failed to create a UDP socket (SIP) on port 5060
Address already in use
```

Figure 8-10 : error after installation

That mean we have to change the port, we changed it to 5080 then it worked as shown in figure 8-11



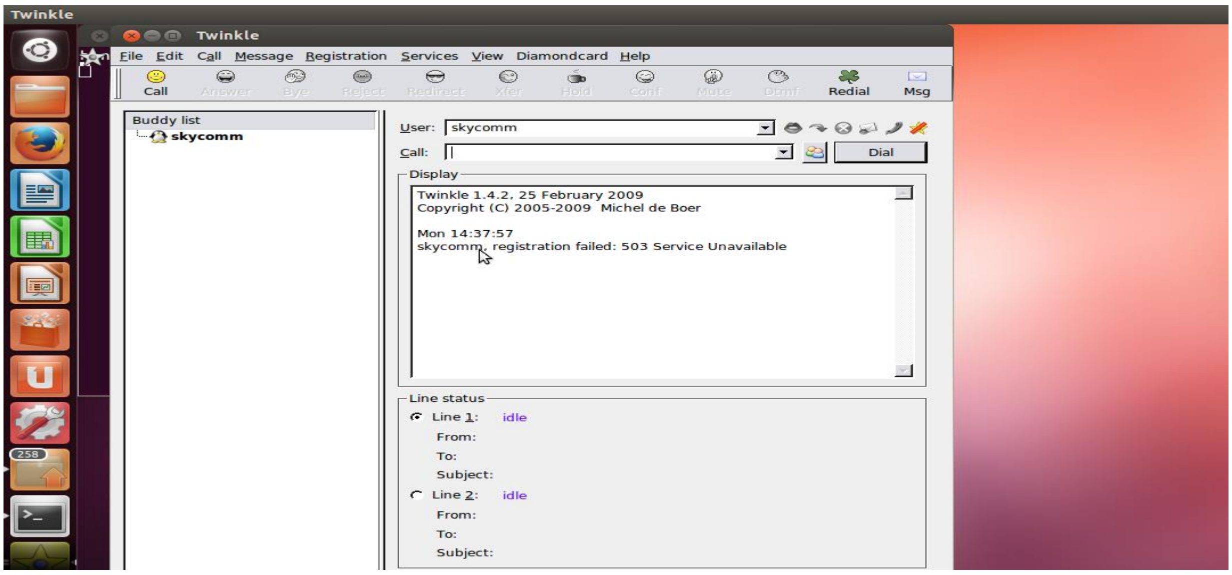
Twinkle - System Settings

Network

SIP port: 5080
RTP port: 8000
Max. SIP message size (UDP): 65535
Max. SIP message size (TCP): 1000000

Figure 8-11 changing SIP port

You can open Twinkle after installation by writing “twinkle” in the terminal, this figure 8-12 shows the window which will appear after opening twinkle.



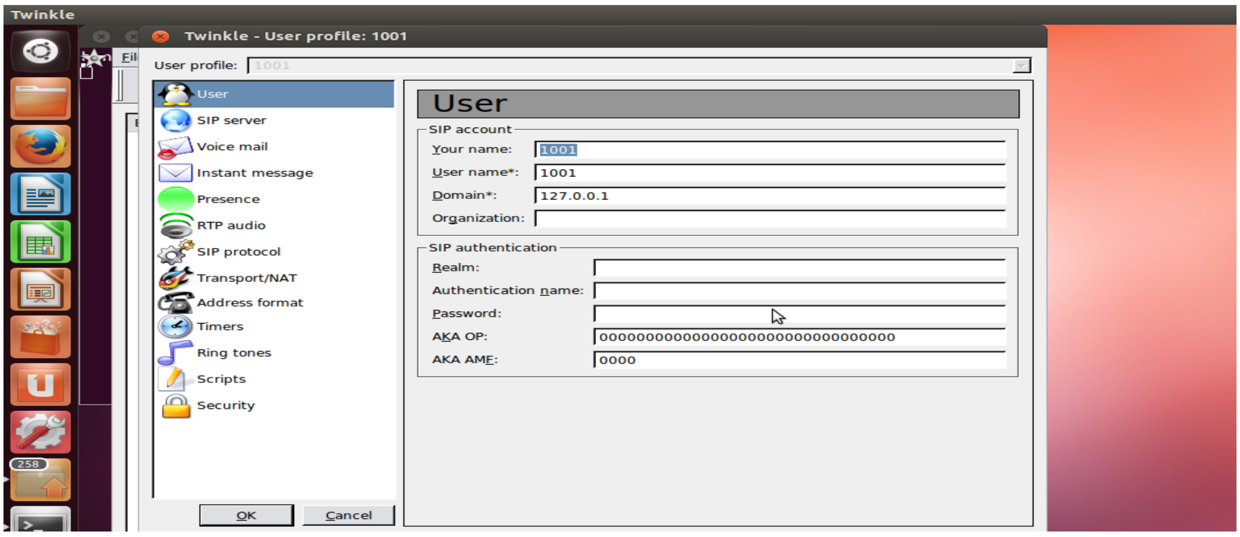
User: skycomm

```
Twinkle 1.4.2, 25 February 2009
Copyright (C) 2005-2009  Michel de Boer

Mon 14:37:57
skycomm, registration failed: 503 Service Unavailable
```

Line 1: idle
From:
To:
Subject:
Line 2: idle
From:
To:
Subject:

Figure 8-12 : Twinkle window

To connect gateway with twinkle, we identify port 1001 as a User as shown in figure 8-13



User profile: 1001

Your name: 1001
User name*: 1001
Domain*: 127.0.0.1
Organization:

Realm:
Authentication name:
Password:
AKA OP: 00000000000000000000000000000000
AKA AMF: 0000

Figure 8-13set 1001 as a user

Then we can make call by Twinkle.

8.6 Gateway with Asterisk

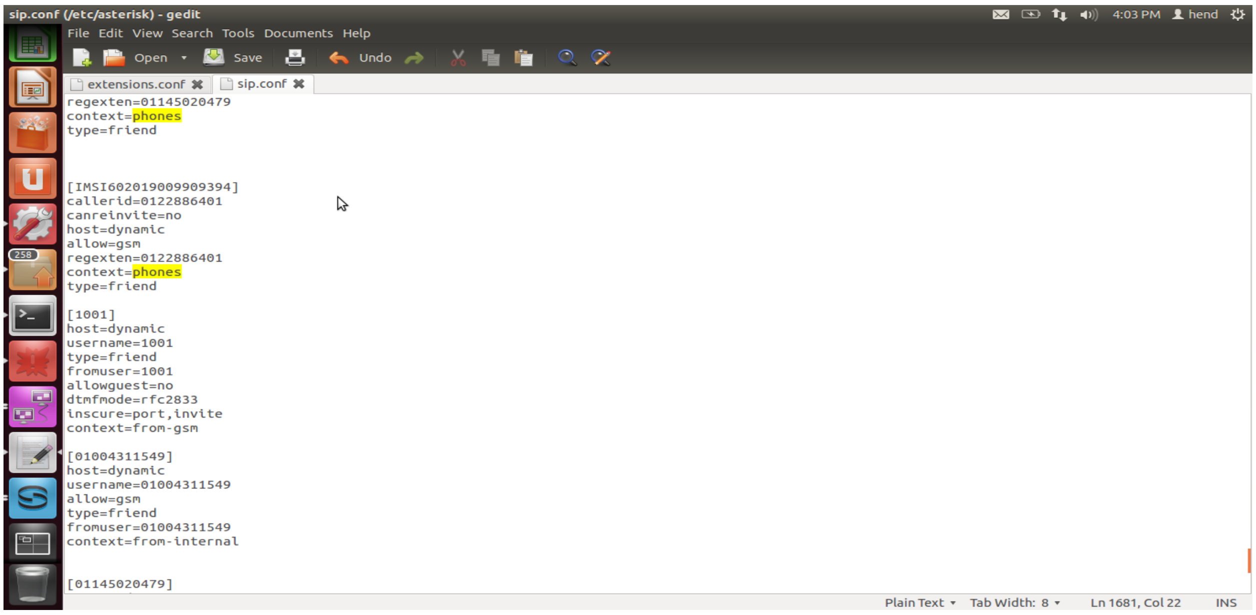
Now, we can easily connect Gateway with Asterisk.

Once we put 1001 IP address the same as IP of your laptop, Asterisk will see the Gateway but we need to add some definitions in sip.conf & extension.conf to can connect them

8.6.1 Sip.conf

In sip.conf we need to add some info to enable us to make a call.

As shown in figure 8-14 We need to add the phone number of the registered handset, define our sip endpoint and our IMSI had already added before.

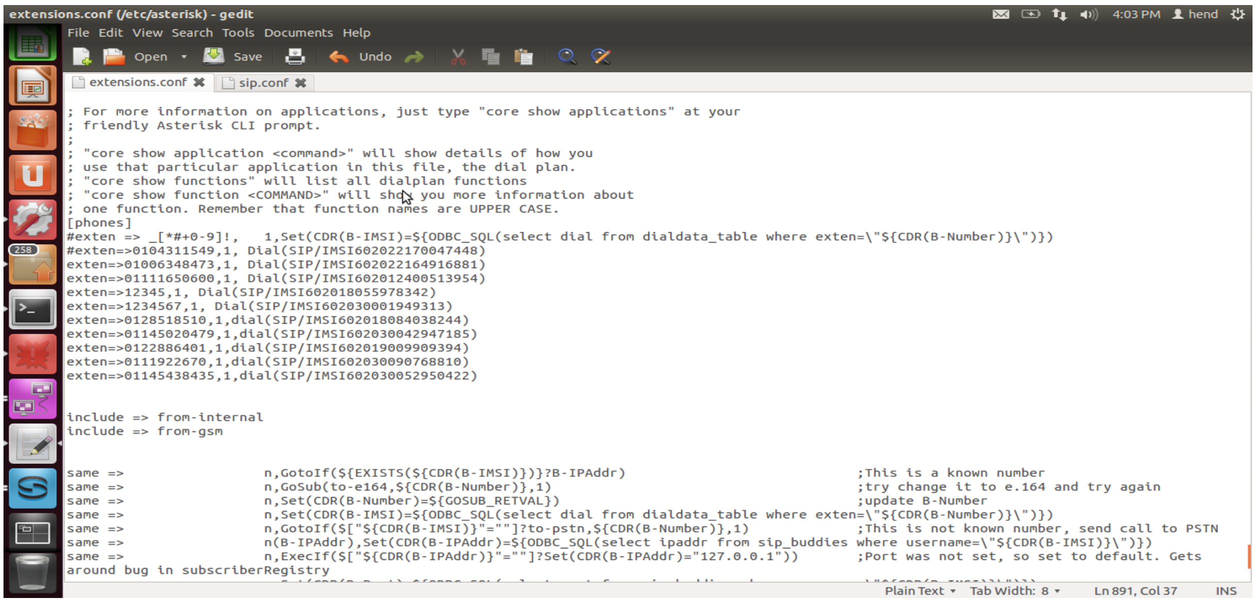


```
regexten=01145020479
context=phones
type=friend



[IMSI602019009909394]
callerid=0122886401
canreinvite=no
host=dynamic
allow=gsm
regexten=0122886401
context=phones
type=friend

[1001]
host=dynamic
username=1001
type=friend
fromuser=1001
allowguest=no
dtmfmode=rfc2833
inscure=port,invite
context=from-gsm

[01004311549]
host=dynamic
username=01004311549
allow=gsm
type=friend
fromuser=01004311549
context=from-internal


[01145020479]
```

Figure 8-14 sip.conf

8.6.2 Extension.conf

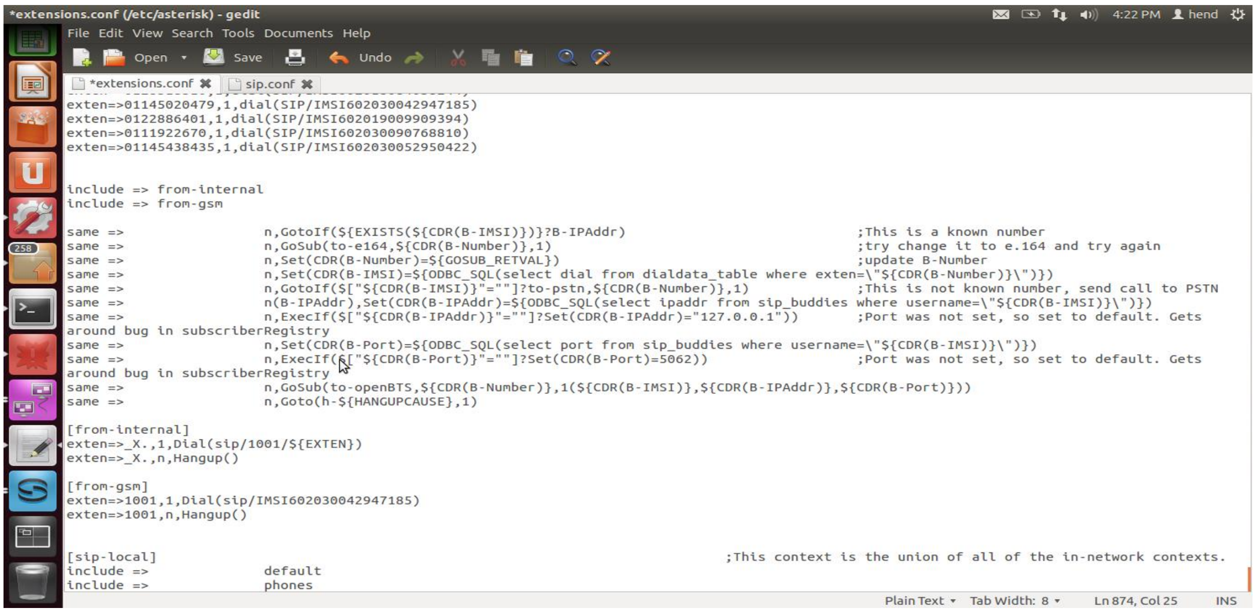
Also, we need to add some info in extension.conf to can make a call like our phones as we added before. As shown in figure 8-15


extensions.conf (/etc/asterisk) - gedit
4:03 PM hend
File Edit View Search Tools Documents Help
Open Save Undo
extensions.conf
sip.conf
Plain Text Tab Width: 8 Ln 891, Col 37 INS

```
; For more information on applications, just type "core show applications" at your
; friendly Asterisk CLI prompt.
;
; "core show application <command>" will show details of how you
; use that particular application in this file, the dial plan.
; "core show functions" will list all dialplan functions
; "core show function <COMMAND>" will show you more information about
; one function. Remember that function names are UPPER CASE.
[phones]
#exten => _[*#+0-9]!,   1,Set(CDR(B-IMSI)=${ODBC_SQL(select dial from dialdata_table where exten=\"${CDR(B-Number)}\")})
#exten=>0104311549,1, Dial(SIP/IMSI602022170047448)
exten=>01006348473,1, Dial(SIP/IMSI602022164916881)
exten=>01111650600,1, Dial(SIP/IMSI602012400513954)
exten=>12345,1, Dial(SIP/IMSI602018055978342)
exten=>1234567,1, Dial(SIP/IMSI602030001949313)
exten=>0128518510,1,dial(SIP/IMSI602018084038244)
exten=>01145020479,1,dial(SIP/IMSI602030042947185)
exten=>0122886401,1,dial(SIP/IMSI602019009909394)
exten=>0111922670,1,dial(SIP/IMSI602030090768810)
exten=>01145438435,1,dial(SIP/IMSI602030052950422)


include => from-internal
include => from-gsm


same =>                 n,GotoIf(${EXISTS(${CDR(B-IMSI)})}?B-IPAddr)                                   ;This is a known number
same =>                 n,GoSub(to-e164,${CDR(B-Number)},1)                                             ;try change it to e.164 and try again
same =>                 n,Set(CDR(B-Number)=${GOSUB_RETVAL})                                            ;update B-Number
same =>                 n,Set(CDR(B-IMSI)=${ODBC_SQL(select dial from dialdata_table where exten=\"${CDR(B-Number)}\")})
same =>                 n,GotoIf($["${CDR(B-IMSI)}"=""]?to-pstn,${CDR(B-Number)},1)                     ;This is not known number, send call to PSTN
same =>                 n(B-IPAddr),Set(CDR(B-IPAddr)=${ODBC_SQL(select ipaddr from sip_buddies where username=\"${CDR(B-IMSI)}\")})
same =>                 n,ExecIf($["${CDR(B-IPAddr)}"=""]?Set(CDR(B-IPAddr)="127.0.0.1"))               ;Port was not set, so set to default. Gets
around bug in subscriberRegistry
```

Figure 8-15 extension.conf(phones)

Don’t forget to include the 2 routing contexts [from-internal] &[from-gsm] like that.as shown in figure 8-16.


*extensions.conf (/etc/asterisk) - gedit
4:22 PM hend
File Edit View Search Tools Documents Help
Open Save Undo
*extensions.conf
sip.conf
Plain Text Tab Width: 8 Ln 874, Col 25 INS

```
exten=>01145020479,1,dial(SIP/IMSI602030042947185)
exten=>0122886401,1,dial(SIP/IMSI602019009909394)
exten=>0111922670,1,dial(SIP/IMSI602030090768810)
exten=>01145438435,1,dial(SIP/IMSI602030052950422)


include => from-internal
include => from-gsm

same =>                 n,GotoIf(${EXISTS(${CDR(B-IMSI)})}?B-IPAddr)                                   ;This is a known number
same =>                 n,GoSub(to-e164,${CDR(B-Number)},1)                                             ;try change it to e.164 and try again
same =>                 n,Set(CDR(B-Number)=${GOSUB_RETVAL})                                            ;update B-Number
same =>                 n,Set(CDR(B-IMSI)=${ODBC_SQL(select dial from dialdata_table where exten=\"${CDR(B-Number)}\")})
same =>                 n,GotoIf($["${CDR(B-IMSI)}"=""]?to-pstn,${CDR(B-Number)},1)                     ;This is not known number, send call to PSTN
same =>                 n(B-IPAddr),Set(CDR(B-IPAddr)=${ODBC_SQL(select ipaddr from sip_buddies where username=\"${CDR(B-IMSI)}\")})
same =>                 n,ExecIf($["${CDR(B-IPAddr)}"=""]?Set(CDR(B-IPAddr)="127.0.0.1"))               ;Port was not set, so set to default. Gets
around bug in subscriberRegistry
same =>                 n,Set(CDR(B-Port)=${ODBC_SQL(select port from sip_buddies where username=\"${CDR(B-IMSI)}\")})
same =>                 n,ExecIf($["${CDR(B-Port)}"=""]?Set(CDR(B-Port)=5062))                          ;Port was not set, so set to default. Gets
around bug in subscriberRegistry
same =>                 n,GoSub(to-openBTS,${CDR(B-Number)},1(${CDR(B-IMSI)},${CDR(B-IPAddr)},${CDR(B-Port)}))
same =>                 n,Goto(h-${HANGUPCAUSE},1)

[from-internal]
exten=>_X.,1,Dial(sip/1001/${EXTEN})
exten=>_X.,n,Hangup()

[from-gsm]
exten=>1001,1,Dial(sip/IMSI602030042947185)
exten=>1001,n,Hangup()


[sip-local]                                                     ;This context is the union of all of the in-network contexts.
include =>              default
include =>              phones
```

Figure 8-16 dial rule (from internal/gsm)

So the call routing will be as follow :

[**from-internal**]

exten => _X.,1,Dial(sip/1001/${EXTEN})

exten => _X.,n,Hangup()

It is for outbound call.For example,when you want to call the number 123456 the call will be sent to the sip trunk 1001.
And will be sent to destination number by gsm gateway with port gsm-1.1.
Here is the call progress:
asterisk server --->1001(sip trunk) --->gsm gateway(gsm-1.1) --->destination number

```
[from-gsm]
 exten => 1001,1,Dial(SIP/IMSI_NUMBER)
 exten => 1001,n,Hangup()
```

This is for inboud call.When someone call the number of gsm-1.1
Here is the call progress:
call ---> gsm-1.1 ---> 1001(sip trunk) --->asterisk server-→ the number which you forward the call to.

8.6.3 Problem

Now, I can call everyone out of my network but when someone wants to call me, there is a problem I will illustrate it.

In our project mobile can see another network called 60208 And I left Etisalat (my operator network) and register manually at my created network 60208

So if anyone in Etisalat for example wants to call me (I am in 60208) but my home network is Etisalat so all my database is in Etisalat we just make a similar but not known network in the country, So Etisalat will search about me in its network, it couldn't find me because I am in 60208 but it can't see our network so Etisalat will tell the number who calls me number busy.

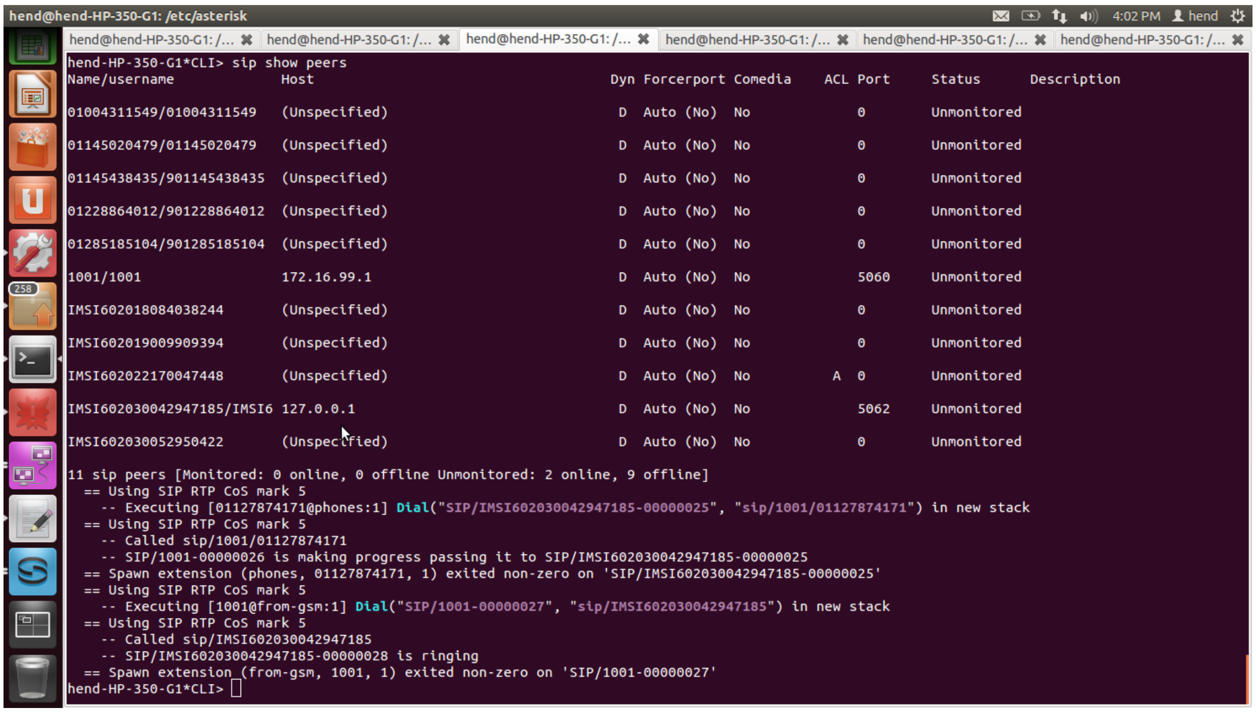
This is the main problem, my target is when someone in any network call anyone in the created network 60208, he can call him.

But cause of this problem and no way operators can see our created network I tried to call the number of GSM 1.1 and routed it to one specific number manually, I could do that successfully.

8.6.4 Final

Briefly, now we can call everyone but when anyone wants to call my network, I forward the call to one specific number which is in airplane. As shown in figure 8-17.

Now I will show you which appears in Asterisk when I make or receive a call.



```
hend-HP-350-G1*CLI> sip show peers
Name/username             Host                                    Dyn Forcerport Comedia    ACL Port     Status      Description

01004311549/01004311549   (Unspecified)                            D  Auto (No)  No             0        Unmonitored

01145020479/01145020479   (Unspecified)                            D  Auto (No)  No             0        Unmonitored

01145438435/901145438435  (Unspecified)                            D  Auto (No)  No             0        Unmonitored

01228864012/901228864012  (Unspecified)                            D  Auto (No)  No             0        Unmonitored

01285185104/901285185104  (Unspecified)                            D  Auto (No)  No             0        Unmonitored

1001/1001                 172.16.99.1                              D  Auto (No)  No             5060     Unmonitored

IMSI602018084038244       (Unspecified)                            D  Auto (No)  No             0        Unmonitored

IMSI602019009909394       (Unspecified)                            D  Auto (No)  No             0        Unmonitored

IMSI602022170047448       (Unspecified)                            D  Auto (No)  No          A  0        Unmonitored

IMSI602030042947185/IMSI6 127.0.0.1                                D  Auto (No)  No             5062     Unmonitored

IMSI602030052950422       (Unspecified)                            D  Auto (No)  No             0        Unmonitored

11 sip peers [Monitored: 0 online, 0 offline Unmonitored: 2 online, 9 offline]
  == Using SIP RTP CoS mark 5
    -- Executing [01127874171@phones:1] Dial("SIP/IMSI602030042947185-00000025", "sip/1001/01127874171") in new stack
  == Using SIP RTP CoS mark 5
    -- Called sip/1001/01127874171
    -- SIP/1001-00000026 is making progress passing it to SIP/IMSI602030042947185-00000025
  == Spawn extension (phones, 01127874171, 1) exited non-zero on 'SIP/IMSI602030042947185-00000025'
  == Using SIP RTP CoS mark 5
    -- Executing [1001@from-gsm:1] Dial("SIP/1001-00000027", "sip/IMSI602030042947185") in new stack
  == Using SIP RTP CoS mark 5
    -- Called sip/IMSI602030042947185
    -- SIP/IMSI602030042947185-00000028 is ringing
  == Spawn extension (from-gsm, 1001, 1) exited non-zero on 'SIP/1001-00000027'
hend-HP-350-G1*CLI> 
```

Figure 8-17 asterisk show

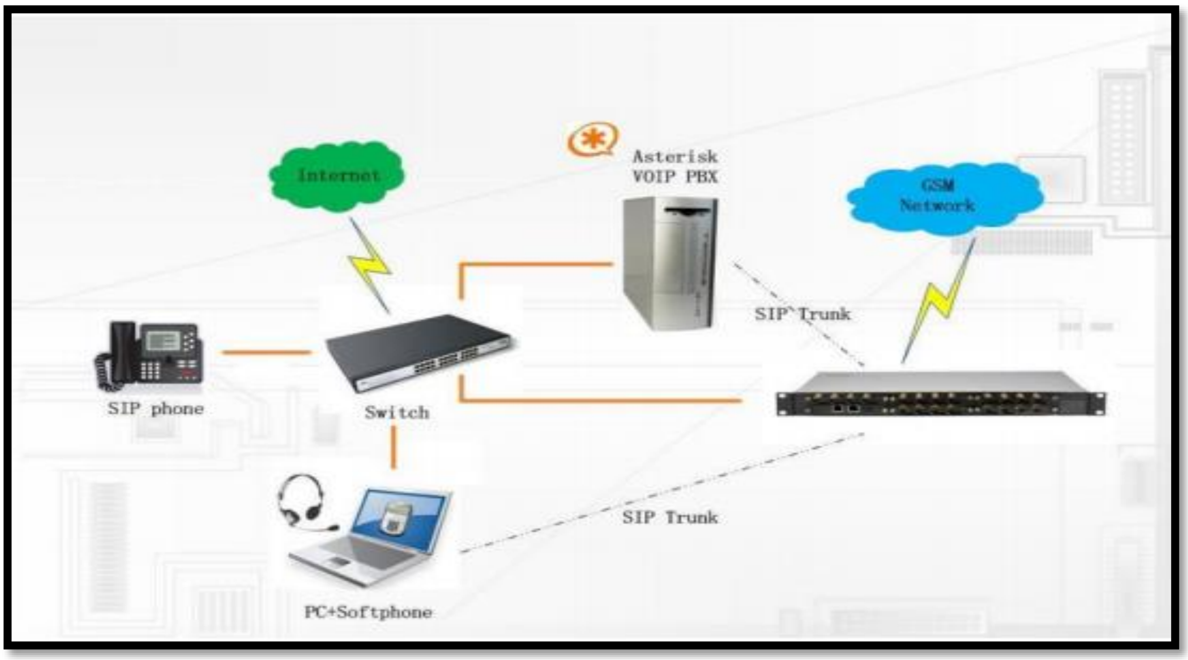


Figure 8-18 Application Diagrams

Chapter 9: Business case model

9.1 Motivation

Marketing for engineering-focused companies is often split into two disciplines: technical marketing and ordinary marketing. Functionally, technical marketing deals with the internal workings parts and how engineers can use them to solve real-world problems. As the name implies, this is a hybrid discipline that requires technical (engineering) excellence and at least rudimentary skills in more general marketing. Ordinary marketing is often focused on finding ways to get people to identify themselves as potential users (buyers) of products.

So we need to be more focus in our field to provide an optimum solution but we should know about some statistic about the companies that support calling service in planes (Technical marketing).

9.2 Statistics & Research

Keeping customers connected to mobile networks in-flight would be a major opportunity for U.S. carriers -- potentially worth $2.4 billion plus a year, according to Akshay Sharma, a wireless network analyst at Gartner.

Now that the federal government is considering an end to its in-flight phone call ban, these but it will cost cell phone companies millions of dollars to install the proper equipment on planes, so analysts expect carriers to recoup those costs with a per-flight fee similar to how in-flight Wi-Fi is used today. Wireless carriers could also charge hefty per-minute voice fees and roaming charges could apply if your cell phone company's network isn't supported on your flight. Companies might finally have a chance to dip into untapped potential revenue.

"*It's a huge coup for the telecom carriers; this opens up a massive market for them*," said Ari Zoldan, CEO of communication technology firm Quantum Networks.

Zoldan's company would be among those retrofitting jets with satellite technology. To keep calls crisp and uninterrupted, the plane would need to be connected constantly, even as it travels at 39,000 feet going 550 miles per hour. That would mean installing a large, powerful, computer-like device that can transmit signals to satellites in space and antennae on the ground.

Phone calls from planes equipped with technology from OnAir, a Swiss company that counts British Airways, Emirates and Singapore Airlines among its customers, is about \$3 to \$4 per minute, said Aurélie Branchereau-Giles, the director of communications for the company. "*You would expect similar pricing*" in the U.S.

Passengers using OnAir's service, as well as AeroMobile, another provider, connect to the cellular network like it's an international network, and are later billed through their wireless carriers.

Many of the people who took to social media to complain about the possibility of having a neighbor barking into a cell phone during a flight would no doubt welcome prohibitively high service costs. As the Journal reports, 51 percent of people have negative feelings about passengers making phone calls on planes, according to a survey cited by the FAA in an advisory group report from earlier this year.

But experts say that in part due to the likely high expense of making a call longer than a few minutes, any fear of having to endure someone else's flight-long phone call is overblown. A picocell system would likely require customers to pay roaming fees for each minute that they are talking on the phone, much like how international roaming is charged today. Virgin Atlantic's AeroMobile service is only available to customers of British carriers O2 and Vodafone and costs £1 per minute for calls and 20 pence for text messages, for example (it's also limited to six users at a time).

Therefore, we want to provide many solutions with market aspect by showing prices of all components of our project.

9.3 System pricing

For any new communication system there is a cost we should calculate it and then decide if we will apply this system or not. So we are going to talk about the price of every component in the system and the number of kits required.

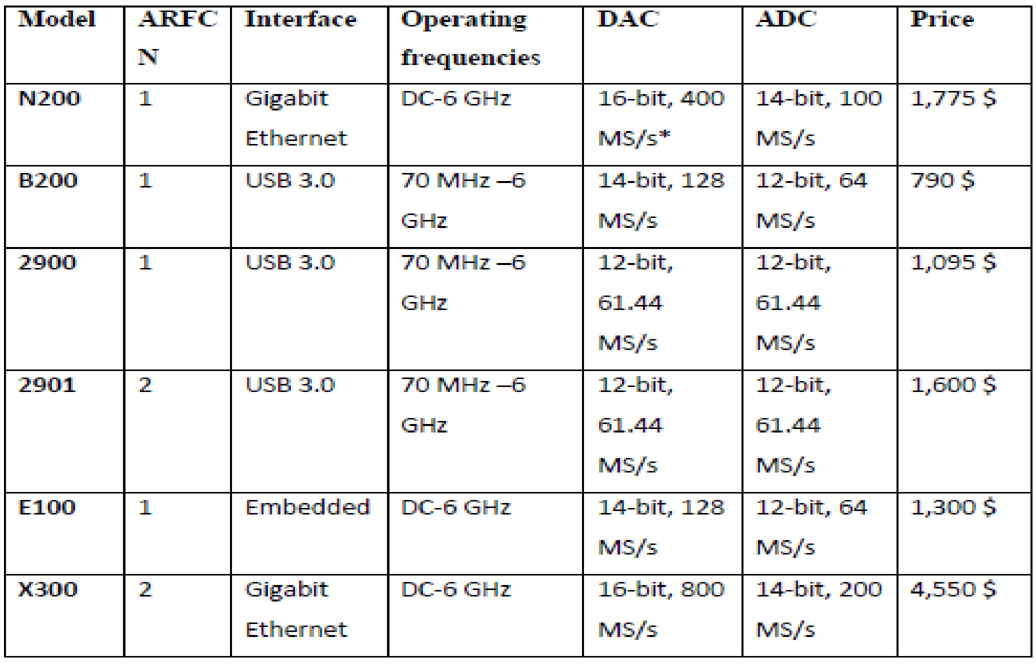
The main components of the system:

1) USRP kit
2) RF daughterboard
3) Antenna
4) Cables
5) Gateway
6) Satellite Data unit

9.3.1 USRP kit

There are many types of USRP kits can be used.

Table 9-1list of available USRPs Prices



Model	ARFCN	Interface	Operating frequencies	DAC	ADC	Price
N200	1	Gigabit Ethernet	DC-6 GHz	16-bit, 400 MS/s*	14-bit, 100 MS/s	1,775 $
B200	1	USB 3.0	70 MHz –6 GHz	14-bit, 128 MS/s	12-bit, 64 MS/s	790 $
2900	1	USB 3.0	70 MHz –6 GHz	12-bit, 61.44 MS/s	12-bit, 61.44 MS/s	1,095 $
2901	2	USB 3.0	70 MHz –6 GHz	12-bit, 61.44 MS/s	12-bit, 61.44 MS/s	1,600 $
E100	1	Embedded	DC-6 GHz	14-bit, 128 MS/s	12-bit, 64 MS/s	1,300 $
X300	2	Gigabit Ethernet	DC-6 GHz	16-bit, 800 MS/s	14-bit, 200 MS/s	4,550 $

9.3.2 RF daughterboard

There are many modes of daughterboard such as receiving mode only or transmitting mode only or full duplex mode.

Table 9-2 list of available Daughterboard prices

Model	Type	Frequency Range	Bandwidth (MHz)	Power output (mW)	Price
LFTX	Rx	0-30 MHz	60	n/a	90 $
Basic RX	Rx	1-250 MHz	100	n/a	90 $
TVRX2	Rx	50-860 MHz	10	n/a	240 $
DBSRX2	Rx	800 MHz – 2.35 GHz	60	n/a	180 $
LFTX	Tx	0-30 MHz	60	1	90 $
Basic TX	Tx	1-250 MHz	100	1	90 $
WBX	Full-Duplex	50 MHz – 2.2 GHz	40	100	565 $
SBX	Full-Duplex	400 MHz – 4 GHz	40	100	565 $
RFX900	Full-Duplex	750-1050 MHz	30	200	300 $
RFX1800	Full-Duplex	1.5 GHz – 2.1 GHz	30	100	300 $
RFX2400	Full-Duplex	2.3 GHz – 2.9 GHz	30	50	300 $

9.3.3 Antennas

There are many types of antennas can be used as we can use directional or omnidirectional antenna. Also we can extend the coverage using leaky feeder.

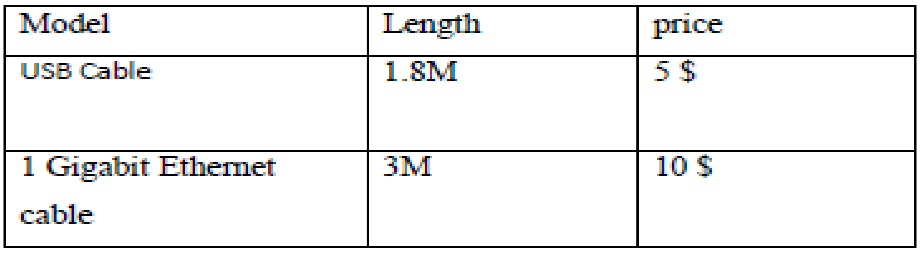
Table 9-3 list of available antennas prices

Model	Antenna type	Price
LP0410	directional antenna	46 $
LP0965	directional antenna	46 $
VERT400	omni-directional vertical antenna	46 $
VERT900	Omni-directional vertical antenna	36 $

9.3.4 Cable

There 2 types of cables to connect the USRP kit to the processing unit:

Table 9-4 list of cables prices



Model	Length	price
USB Cable	1.8M	5 $
1 Gigabit Ethernet cable	3M	10 $

9.3.5 Gateways

There are many types of GSM gateways can be used as IP network to connect between different networks.

Table 9-5l List of GSM GATEWAYS prices

Model	price
VS-GW2120 GSM	$749
VS-GW1600 GSM	$449
VS-GW1202 GSM	$399
WGW1002G	$250

9.4 System Capacity

To calculate the capacity (number of Kits used in the system), firstly we have to know:

1- The aircraft dimensions and number of passengers.

2- Erlang B table.

9.4.1 The aircraft dimensions and number of passengers

Airbus A380-800 -as shown in the figures 9-1 and 9-2 below -is the world's largest passenger airliner –we make our calculations depending on it–, and the airports at which it operates have upgraded facilities to accommodate it. And it provides seating for 525 people in a typical three-class configuration or up to 853 people in an all-economy class configuration. And there is another aspect we need to know the dimensions of the aircraft (length=72.7m and width=79.7).

Figure 9-1Airbus A380-800

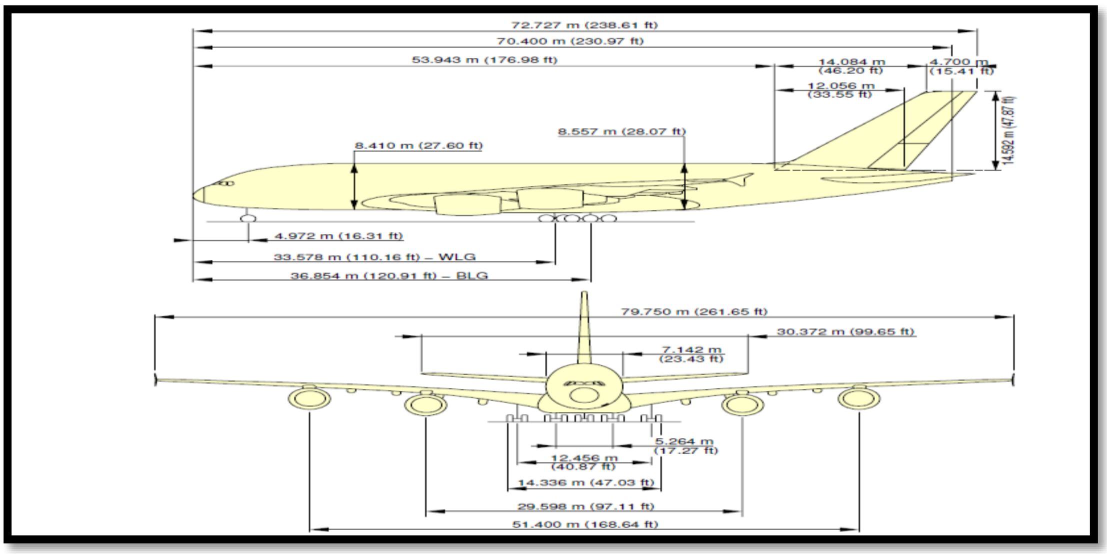


Figure 9-2Airbus A380 Dimensions

9.4.2 Erlang B table

The Erlang (symbol E) is a dimensionless unit that is used in telephony as a measure of offered load or carried load on service-providing elements such as telephone circuits or telephone switching equipment.

It depends on:

1- Blocking probability: it is the fraction of time a call request is denied because all channels are busy. This probability is usually specified for a given system. It is typically desired to be 3%.

2- Number of ARFCN

By determining the number of ARFCN and blocking probability, we can calculate how many Erlang the system can support in which we can determine the system capacity.

9.5 Determining number of ARFCNs (USRP kits)

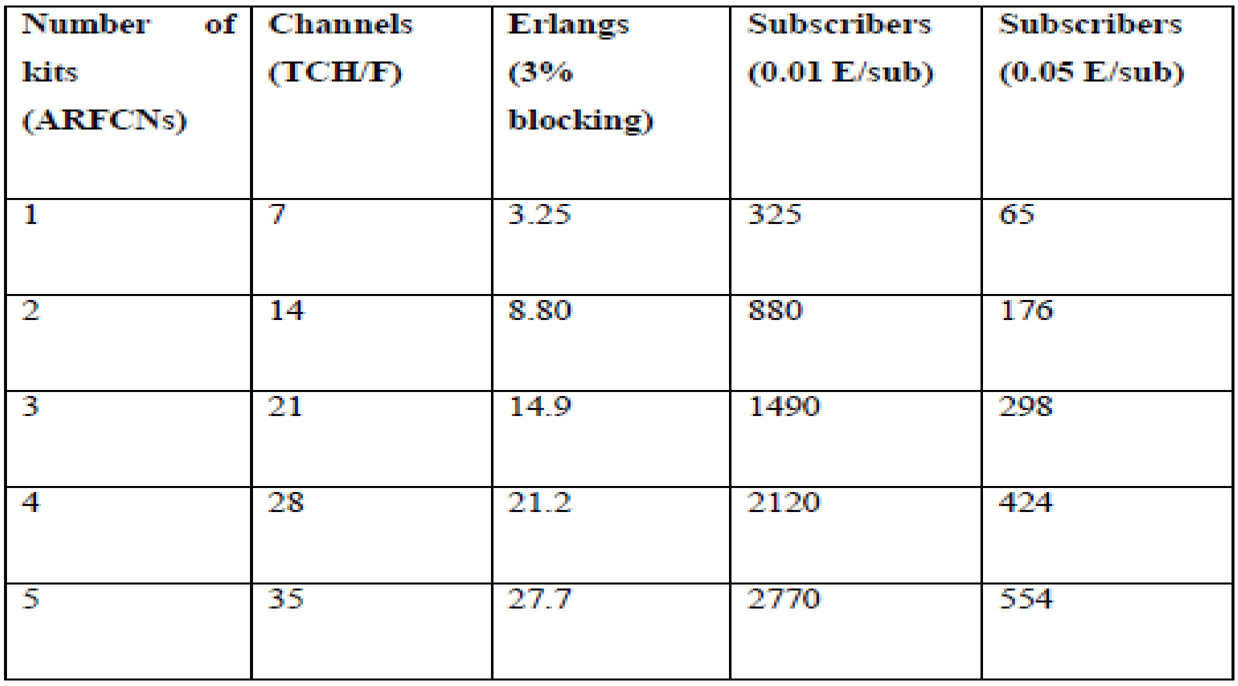
After knowing the dimensions and number of passengers of the largest aircraft and using the Erlang tables we have found that if we take only in our consideration the dimensions of the plane (length= 72m) we will need only 3 kits as the coverage radius of VERT900= 15m, but if we take on our consideration the number of passengers on the plane, we should determine the erlang per subscriber –with blocking probability= 3%– to know the suitable number of kits which illustrated on table 8.5.

We take two cases to determine number of kits depending on number of erlangs per subscribers:

1st case: each subscriber has 0.01 E

2nd case: each subscriber has 0.05 E

Table 9-6number of kits and number of subscribers with fixed Erlang



Number of kits (ARFCNs)	Channels (TCH/F)	Erlangs (3% blocking)	Subscribers (0.01 E/sub)	Subscribers (0.05 E/sub)
1	7	3.25	325	65
2	14	8.80	880	176
3	21	14.9	1490	298
4	28	21.2	2120	424
5	35	27.7	2770	554

In the first case we can have only 2 kits but due to the length (75 m) we should have 3 kits but in the second case the least number of kits we can have is 5 according to the number of subscribers (number of passengers onboard).

We have discussed before in this chapter the solution using USRP kit and its components there are another method that provide the solution like Pico cells & femtocells and also there are used by airlines companies.

Chapter 10: Future work

10.1 Connect our network with satellite link

In our project we can make calls and send massages. When plane is flying, connection with satellite must be exist to receive the request of making calls or sending massages and look for the desired person and make the operation of calls or massages successfully.

As in Chapter 4: So we will talk about the mechanism of calling between two mobile phones one of them in the plane and the other is not and we will identify the different phases and problems that face us to make the two mobile phones are attached to each other and how calling process is done successfully.

10.2 Using GPRS services

In this project we can apply data by GPRS services then we can use internet on the plane.

10.2.1 General Packet Radio Service

GPRS was one of the earliest cell phone data access technologies. It stands for General Packet Radio Service, It is a non-voice wireless Internet technology that is very popular due to the fact that it can support both phone calls and Internet data transmission. It is a second generation (2G) and third generation (3G) or sometimes referred to as in-between both generations, 2.5G wireless data service that extends GSM data capabilities for Internet access, multimedia messaging services, and early mobile Internet applications via the wireless application protocol (WAP),as well as other wireless data service.

With GPRS technology offering data services with data rates up to a maximum of 172 kbps, facilities such as web browsing and other services requiring data transfer became possible. Although some data could be transferred using GSM, the rate was too slow for real data applications.

GPRS technology brings a number of benefits for users and network operators alike over the basic GSM system. It was widely deployed to provide a realistic data capability via cellular telecommunications technology.

It offers some benefits when it was launched like:

1) speed: that it offers a much higher data rate than was possible with GSM. Rates up to 172 kbps are possible, although the maximum data rates realistically achievable under most conditions will be in the range 15 - 40 kbps.
2) Packet switched operation: Unlike GSM which was used circuit switched techniques, GPRS technology uses packet switching, this makes far more efficient use of the available capacity, and it allows greater commonality with Internet techniques.

It is important to know that the GSM and GPRS elements of the system operate separately. The GSM technology still carries the voice calls, while GPRS technology is used for the data. As a result voice and data can be sent and received simultaneously.

10.2.2 How Does GPRS Work?

In GSM cell phone systems, there will always be idle radio capacity. This is the capacity of a network provider that is not being used and it stays unused until other cell phone users decide to make phone calls. GPRS uses this idle radio capacity to establish a data network to be used for data transmission. If a network provider's idle radio capacity decreases, which means a lot of phone calls are being serviced, data transmission and speed decreases as well. Cell phone calls have a higher priority than Internet data transmission in cell phone network providers.

Due to its packet-based nature, which means it transmits data as a series of data packets routed through multiple paths across a GSM network, GPRS does not require a network to have continuous data transmission. This data transmission can share network radio channels easily and efficiently. These network radio channels have 8 time slots each and their maximum data transmission is 13.4 kbps. Every time slot that is not being used can be used by GPRS to transmit data packets. In a best case scenario, user can experience a GPRS data rate of approximately 170 kbps per network radio channel. In this scenario, only data transmissions are using the network channels without voice calls hogging the network.

GPRS technology is an elegant solution for network providers to offer additional services with minimal effort. Cell phone network providers only need to set up a few new infrastructure nodes plus some software upgrades while still using their existing GSM network infrastructure to add an overlay GPRS network to their cell phone service.

10.2.3 What is the relation between GPRS and GPS?

GPRS is technology that many GPS tracking devices are utilizing in order to get up to the minute information with tracking. Once the GPS device records the data it can then be transmitted through GPRS to another central location such as a computer or through an e-mail. It is the GPRS technology that allows for real time updates to

GPS tracking systems. It is this direct GPRS connection that gives the user of the GPS system the most reliable data available on the market today.

10.2.4 Is that possible to use GPRS service in openbts for wireless data connection?

Yes, it is possible and it can be done by enabling the GPRS service. Additional components are not needed only some configurations must be placed on your Linux host for things to work correctly.

10.3 Upgrade to 3G

Our project is based on GSM network in future work we can install the additional components for 3G and 4G and use it. but 3G we need to know the spreading code and the spreading process so it is more complicated. We need to make deals with the operators to be able to use other network properly.

Chapter 11: Conclusion

To wrap up things, Software Defined Radio has been an interesting and concerning topic to a lot of researchers and engineers, with the exponential growth in the ways and means by which people need to communicate, data communication, voice communication, and video communication. In addition, it modifies the radio devices easily and lowers the cost which has become one of the most critical point in business. It also brings flexibility and less power consumption. In this bachelor thesis we used these capabilities of the Software Defined Radio to implement our communication system during flight.

Firstly, we started to get acquainted and familiarized with the SDR by reading its main components and block diagrams from outside and decided to see which of the components would suit us. We mainly chose the B100 USRP as our platform, it was chosen according to many measures which was discussed previously. Moreover, we chose also WBX as our daughter board as it supported the frequency of GSM and it was the available one for us. After that we started to read and get familiar with the Linux Ubuntu commands and how to deal with it. Adding to that, we have read about the frequencies used in the navigation system of the plane to assure safety and not to allow any interference and we found there is no safety issues or problems that would happen. Moreover, the length of the plane and maximum users to know the number of USRPs kits we need and to know the coverage area we will need. Moreover, we decided to use the GSM at 900 MHZ due to an accurate study discussed before. After becoming more knowledgeable about the tools, technologies and the USRPB100 we started the second step.

The second step was reading and making a deep research about the SkyComm solution. Which was divided into many blocks. One of these blocks was the leaky feeder, which is an adaptation of the standard coaxial cable, with one key difference; the outer conductor is slotted and punctured allowing the cable to radiate. These cables are installed above the ceiling panels along the whole aircraft at very low radiation power level. The second block was the ANC, it generates a broadband noise floor which is being emitted through existing leaky line antenna masking reception, they measure and ensure that handsets can only connect to on board GSM network and will then operate with the lowest possible transmission power level GSM-1800 power control level its nominal output power of 0 dbm, which is used for safety issues on the plane. Adding to that the satellite unit which is a ready installed device in the aircraft which is responsible for air/ground communication this part we had to study that is it feasible to connect the USRP to this block and we found that we are able to connect easily to it. Afterwards the call will go in a full duplex way from satellite to earth station and vice versa. We were afraid from a problem which was the Doppler effect during call but after research we found that no Doppler effect will happen as the USRP is in the plane has the same speed of plane and from the point of view of the satellite it is already used for the navigation of the plane so no problem for this part. After getting deep in the solution we had to apply what we have learned and searched in real and try to simulate the solution in real so we started the third phase.

The third phase was installing the system on the USRP B100 and making our internal network by making two mobiles communicate with each other but both must be connected to the USRP which was the hard part as there was a very scarce sources about the information and documentation. We started in implementing the GSM Base station using SDR kit and GNU radio. Throughout the implementation process, we have learned how to alter some GSM parameters such as the ARFCN, allocation of phone numbers, adding mobile to the network and frequency which was 900 MHZ. This phase was completely a success and we reached our goal which is covering about 15 m radius in the lab which was not like the theoretical measure 25 m as there was many obstacles in the lab and we were allowed to perform a call, sending and receiving an SMS between two mobiles but both mobiles must be connected to the same USRP, as said before that both mobiles must be connected to the same network. In this part we were able to make 3 calls simultaneously and non-limited number in sending a message. After passing this phase in a complete success we had to make the final phase.

Finally, we want to connect between our network and other networks, to do that we used GSM gateway which enables us to achieve the interconnection between networks. We configured it with configuration settings which achieved our target then connect it to our network. Now anyone from our network can call anyone outside whether their network through gateway.

References

[1] Vasco Pereira and Tiago Sousa, "*Evolution of Mobile Communications: from 1G to 4G*," IEEE Communications Magazine, July. 2004.

[2] Tachikawa, Keiji, "*A perspective on the Evolution of Mobile Communications*",IEEE Communications Magazine, October 2003, pp. 66-73.

[3] "*OFDM implementation in communication system using USRP N210*", first Edition, GUC, June 4, 2013.

[4] "*Communication System Using USRP N210*", first Edition, GUC, June 4, 2013.

[5] Range Networks, Inc., "*OpenBTS Application Suite. User Manual*," April, 15.2014.

[6] Leaky feeder. Retrieved from: https://en.wikipedia.org/wiki/Leaky_feeder

[7] W.L. Gore & Associates, Inc. "*Leaky Feeder Antennas for Airborne Wi-Fi*,"Microwave Journal, October 15, 2013.

[8] Dennis Roddy, *Satellite communication*, 3rd Edition. McGraw-Hill, 2001.

[9] M.Werner, M.Holzbock, "*Aeronautical broad band communication via satellite*,"

[10] Ground Station. Retrieved from: https://en.wikipedia.org/wiki/Ground_station

[11] J.C. Palais, "*Fiber Optic Communications*" Fifth Edition, Prentice Hall, 2005

[12] Michael Iadema, Range Networks, Inc., "*Getting Started with OpenBTS*,"

[13] http://www.twinklephone.com/

[14] https://www.ettus.com/product/

[15] https://en.wikipedia.org/wiki/Erlang_(unit)

[16] http://openbts.org/get-the-code/

[17] http://blogs.digium.com/2012/11/14/how-to-install-asterisk-11-on-ubuntu-12-4-lts/

[18] http://gnuradio.org/redmine/projects/gnuradio/wiki/UbuntuInstall

[19] http://gnuradioc.blogspot.com.eg/2013/06/how-to-install-gnuradio-on-ubuntu-1204_26.html

Appendix A: Ubuntu 12.04 installation

To install Ubuntu – first create a live USB, CD or DVD installer and boot your system – you may need to change your BIOS settings if your computer doesn't boot from the live media installer. To get into the BIOS menu simply hit F2 or F12 or Del key – it depends on your system. Then, navigate through the arrow key and go boot devices/options and enable the USB boot, then change the order of bootable media (put USB on top) and save it (you may have to hit F10 for that).

Step1. Create USB installer:

Download Ubuntu 12.04 ISO (according to your computer architecture such as 32 bit or 64 bit, and of course the Desktop Edition)
If you're using Windows – then create a USB installer using the application – Universal USB Installer. It's quite easy – download and run this installer – and locate the ISO file, select the target USB drive. If you're using any older version of Ubuntu such as 11.10 or 10.04 then there is one application you need to use – Startup Disk Creator, installed by default on Ubuntu. Run the application, locate the ISO image and make your pen-drive bootable.
Anyway, at the end of this step – you must have a USB stick installer for Ubuntu 12.04.

Step 2. Initiating Installation procedure*:*

Now, you got the live usb installer for Ubuntu 12.04 LTS, restart your computer and boot it from the installer. If there is any booting problem check your BIOS settings – it should be able to boot from a USB and the priority should be #1 (although it's not really required if you manually select the bootable media).First look at Ubuntu 12.04.

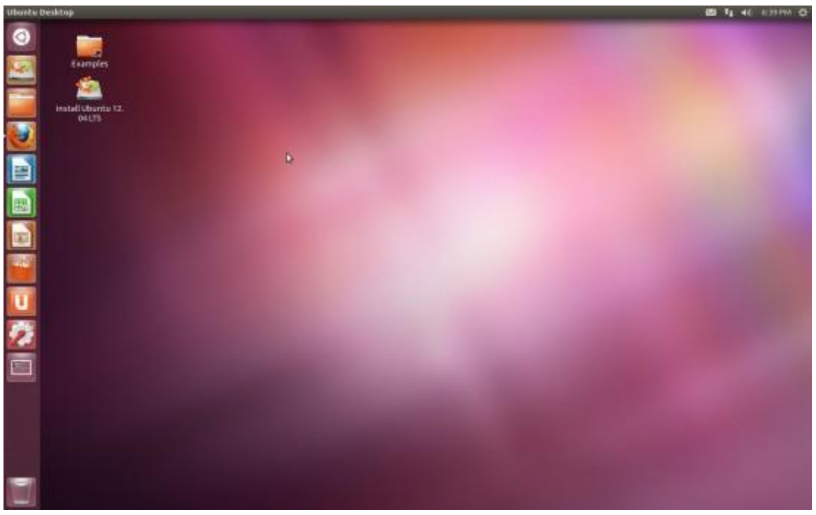


Figure A-1 Ubuntu 12.04 desktop

Click on *Install Ubuntu 12.04 LTS* button to start the installation process – although you can get an overview without even installing it on your hard-drive.

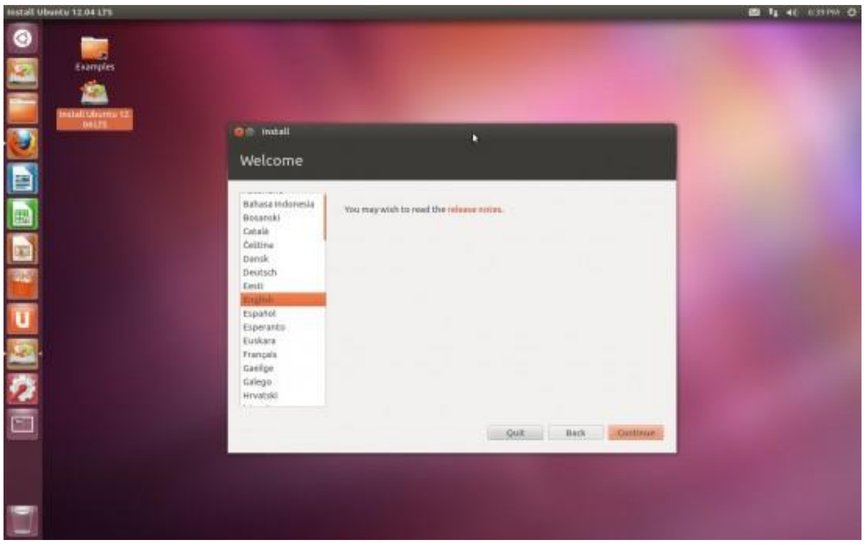
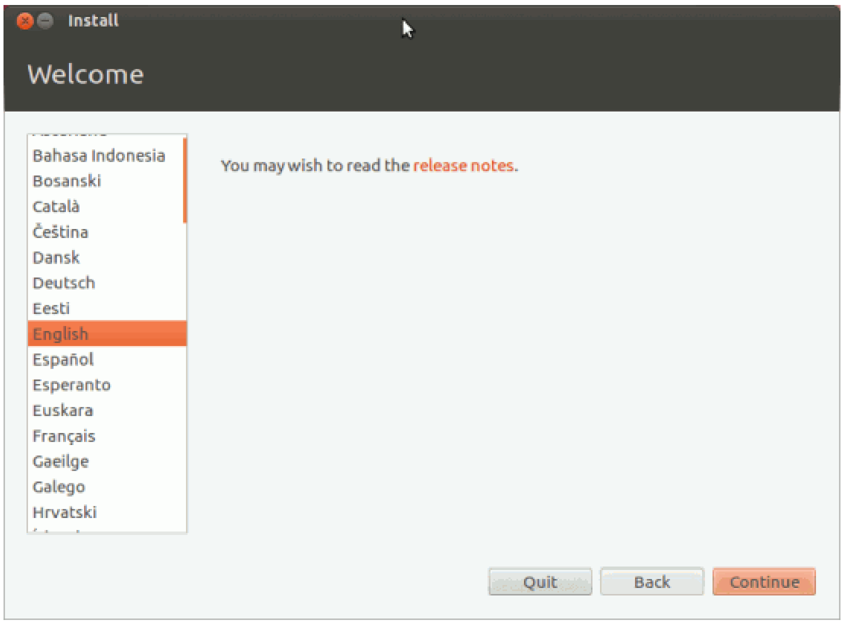


Figure A-2: start installing Ubuntu

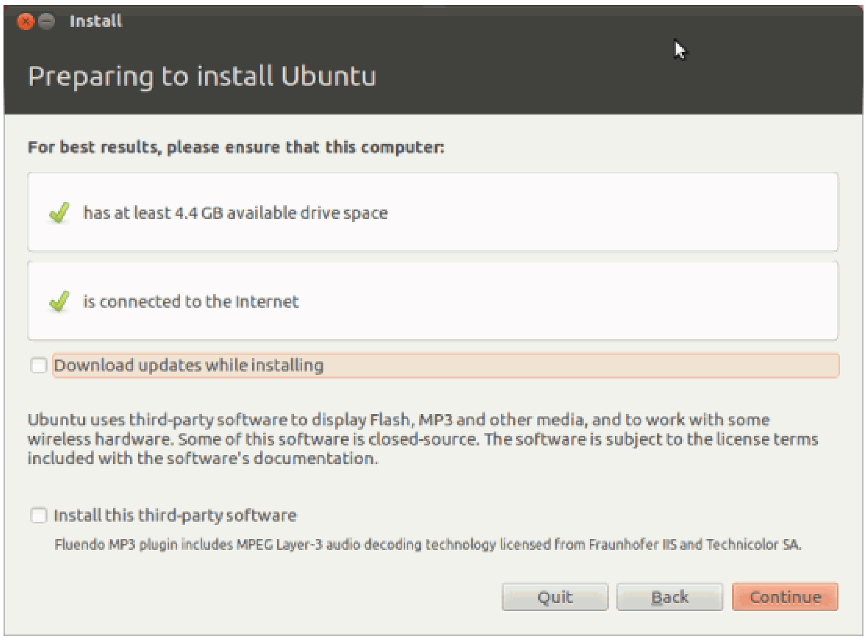
Now select Language – The default is English, but Ubuntu supports a lot of other languages – so you can run Ubuntu in your native language.



You may wish to read the release notes.

Figure A-3: select language

Now, you're ready to install Ubuntu 12.04 on your hard disk. Although Internet connection is recommended but it's not strictly required during installation as you can always updates or install new packages later. In some cases – Wi-Fi network it may not be detected automatically – just relax you can configure that later.


Install

Preparing to install Ubuntu

For best results, please ensure that this computer:

✓ has at least 4.4 GB available drive space

✓ is connected to the Internet

☐ Download updates while installing

Ubuntu uses third-party software to display Flash, MP3 and other media, and to work with some wireless hardware. Some of this software is closed-source. The software is subject to the license terms included with the software's documentation.

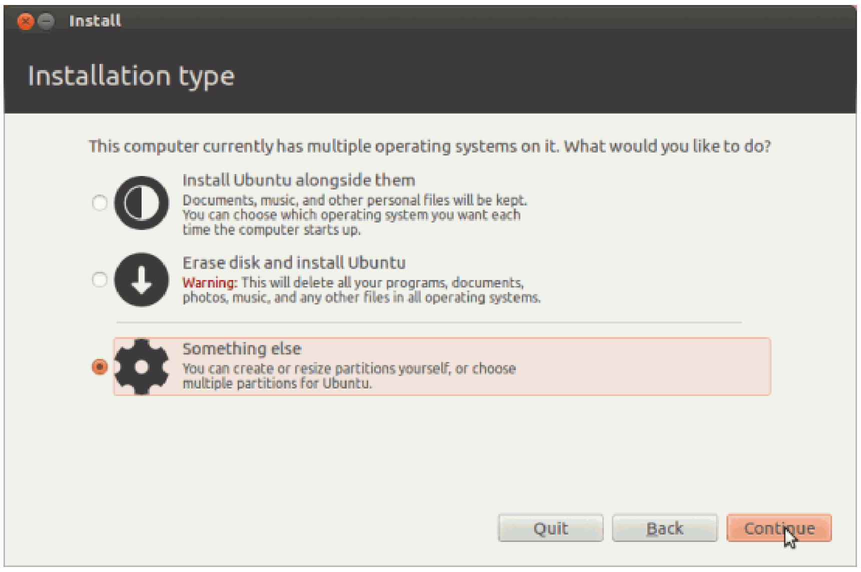
☐ Install this third-party software

Fluendo MP3 plugin includes MPEG Layer-3 audio decoding technology licensed from Fraunhofer IIS and Technicolor SA.

Figure A-4: preparing to install Ubuntu

Step 3. Create hard drive partition for Ubuntu:

Select Installation type – ‘something else’, the third option as it’s the safest method and you get more control over what’s going on.


Install

Installation type

This computer currently has multiple operating systems on it. What would you like to do?

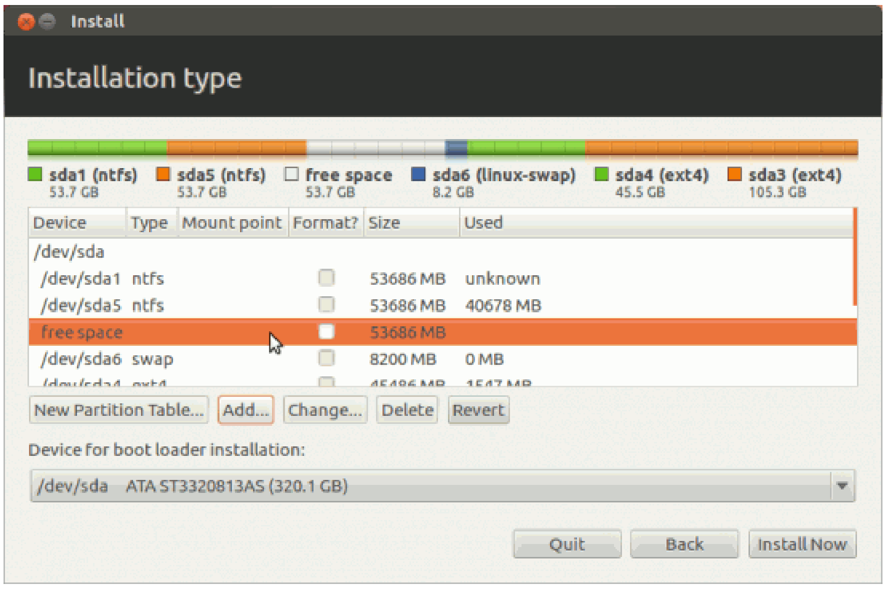
○ Install Ubuntu alongside them
Documents, music, and other personal files will be kept. You can choose which operating system you want each time the computer starts up.

○ Erase disk and install Ubuntu
Warning: This will delete all your programs, documents, photos, music, and any other files in all operating systems.

● Something else
You can create or resize partitions yourself, or choose multiple partitions for Ubuntu.

Figure A-5: installation type

Now you need to create an ext4 partition for Ubuntu 12.04; use the partition manager – swap out Windows (if you don’t want to keep windows) or unused partition or create new partition from existing hard drives.



🟩 sda1 (ntfs) 53.7 GB · 🟧 sda5 (ntfs) 53.7 GB · ⬜ free space 53.7 GB · 🟦 sda6 (linux-swap) 8.2 GB · 🟩 sda4 (ext4) 45.5 GB · 🟧 sda3 (ext4) 105.3 GB

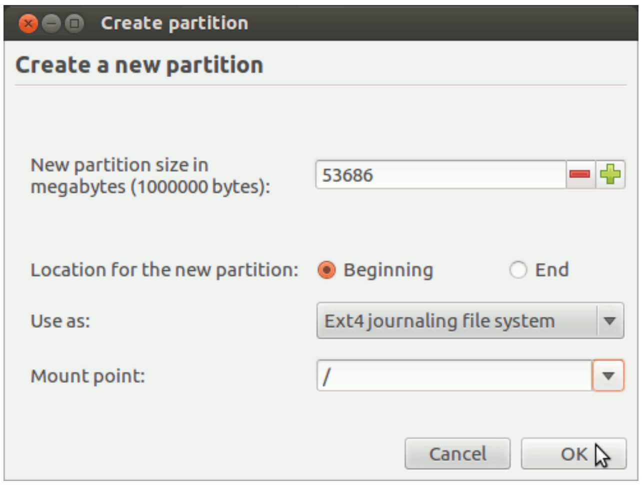
Device	Type	Mount point	Format?	Size	Used
/dev/sda					
/dev/sda1	ntfs		☐	53686 MB	unknown
/dev/sda5	ntfs		☐	53686 MB	40678 MB
free space			☐	53686 MB	
/dev/sda6	swap		☐	8200 MB	0 MB
/dev/sda4	ext4		☐	45486 MB	1547 MB

Device for boot loader installation:

/dev/sda ATA ST3320813AS (320.1 GB)

Figure A-6: Select free space for Ubuntu

Now click on *Free Space* -> *Add* -> *Create Partition*.



Create a new partition

New partition size in megabytes (1000000 bytes): 53686

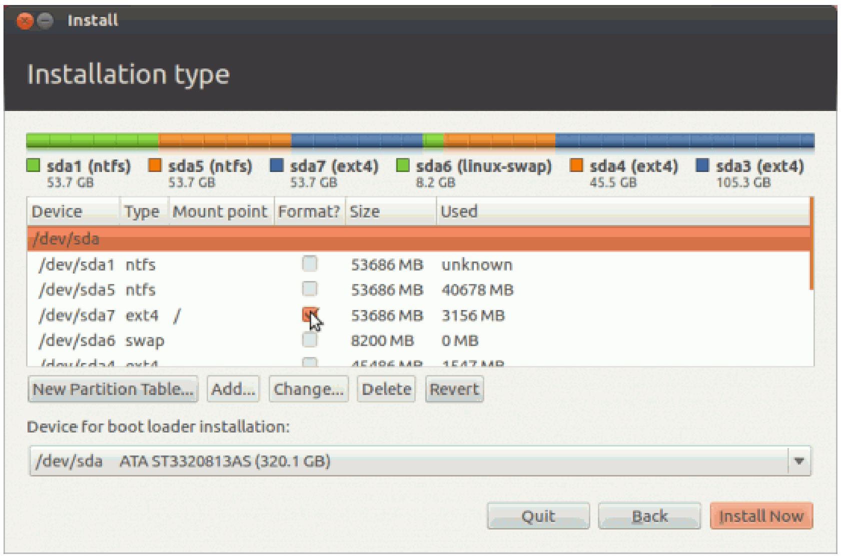
Location for the new partition: [✓ Beginning] End

Use as: Ext4 journaling file system

Mount point: /

Figure A-7: Create partition

select the target partition (check the box), where Ubuntu 12.04 will be installed.



🟩 sda1 (ntfs) 53.7 GB 🟧 sda5 (ntfs) 53.7 GB 🟦 sda7 (ext4) 53.7 GB 🟩 sda6 (linux-swap) 8.2 GB 🟧 sda4 (ext4) 45.5 GB 🟦 sda3 (ext4) 105.3 GB

Device	Type	Mount point	Format?	Size	Used
/dev/sda					
/dev/sda1	ntfs		☐	53686 MB	unknown
/dev/sda5	ntfs		☐	53686 MB	40678 MB
/dev/sda7	ext4	/	✓	53686 MB	3156 MB
/dev/sda6	swap		☐	8200 MB	0 MB
/dev/sda4	ext4		☐	45486 MB	1547 MB

Device for boot loader installation:

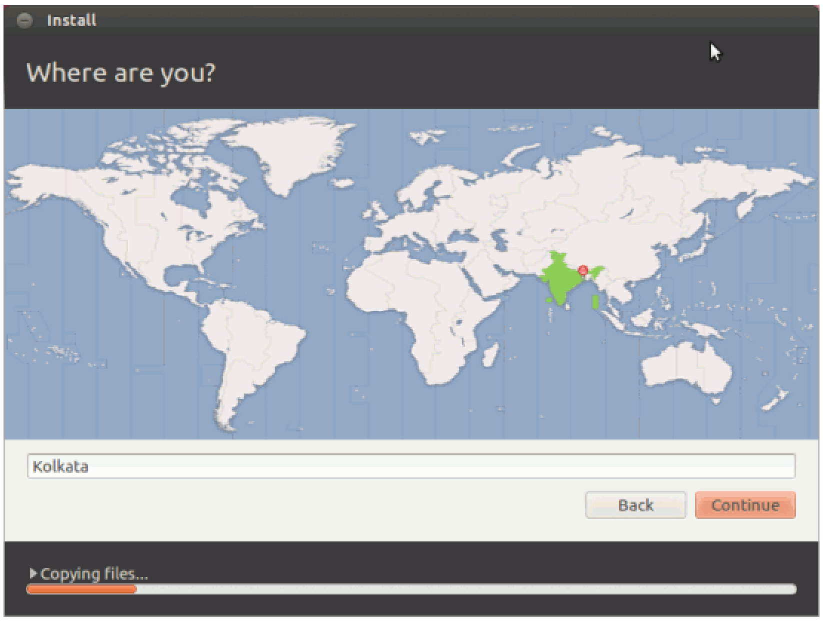
/dev/sda ATA ST3320813AS (320.1 GB)

Figure A-8: partition table after adding Ubuntu

Until now, no hard disk changes has been committed – Now you can click on install now button to begin installation.

Step 4. Installing Ubuntu 12.04:

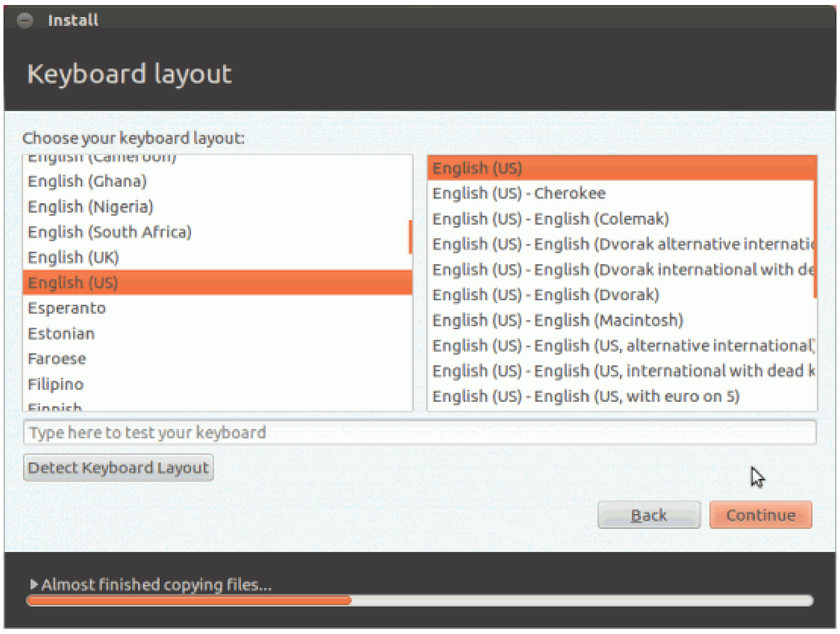
Select your physical location from Map.



Kolkata

Figure A-9: Select Region

Next, choose the keyboard layout, if you’re not sure what option you should choose – then you should probably go for default value.


Install

Keyboard layout

Choose your keyboard layout:

English (Ghana)	English (US) - Cherokee
English (Nigeria)	English (US) - English (Colemak)
English (South Africa)	English (US) - English (Dvorak alternative internati
English (UK)	English (US) - English (Dvorak international with de
[✓ English (US)]	[✓ English (US)]
Esperanto	English (US) - English (Dvorak)
Estonian	English (US) - English (Macintosh)
Faroese	English (US) - English (US, alternative international)
Filipino	English (US) - English (US, international with dead k
	English (US) - English (US, with euro on 5)

Type here to test your keyboard

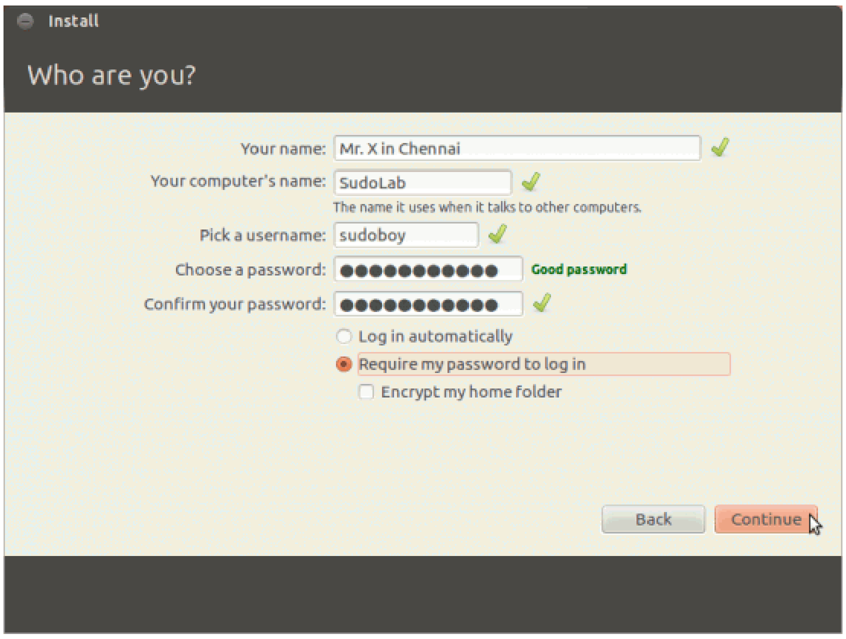
Detect Keyboard Layout

Back Continue

Almost finished copying files...

Figure A-10: Keyboard layout

Enter the user (You, the sudo user) account detail – that will be created during installation procedure (you can create/manager user later).


Install

Who are you?

Your name: Mr. X in Chennai ✓

Your computer's name: SudoLab ✓

The name it uses when it talks to other computers.

Pick a username: sudoboy ✓

Choose a password: ●●●●●●●●●●● Good password

Confirm your password: ●●●●●●●●●●● ✓

○ Log in automatically

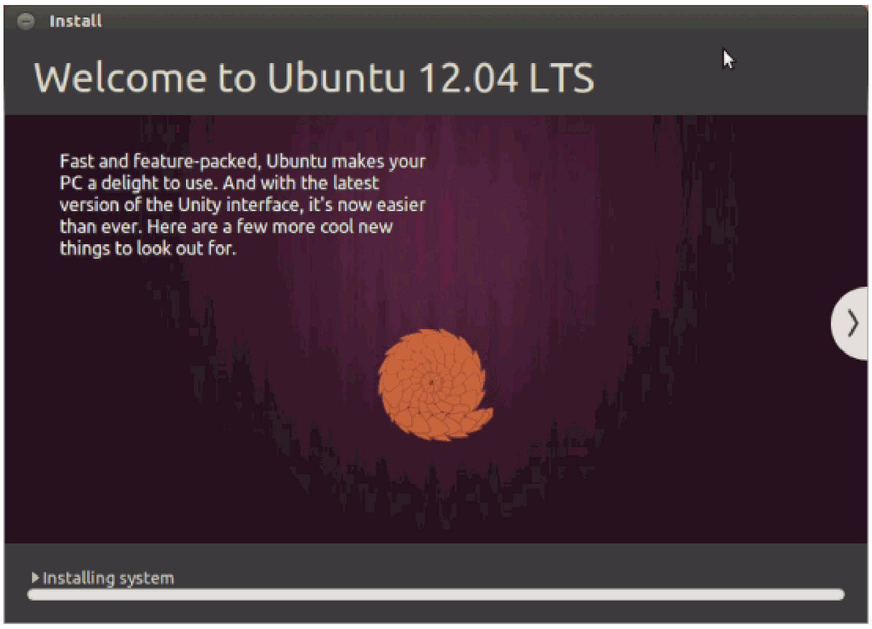
[✓ Require my password to log in]

☐ Encrypt my home folder

Back Continue

Figure A-11: personal information

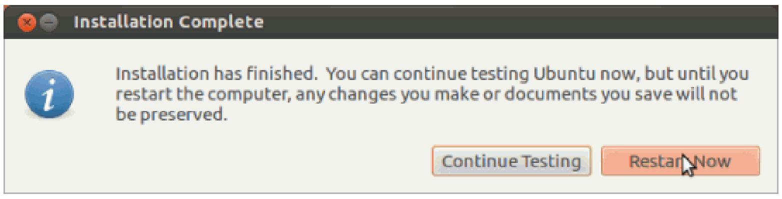
Wait until the installation process is complete. If you're connected to internet – it may take little extra time because of update/extra language packs – if you don't want them, then skip those steps.



Fast and feature-packed, Ubuntu makes your PC a delight to use. And with the latest version of the Unity interface, it's now easier than ever. Here are a few more cool new things to look out for.

Figure A-12: installing system

Step 5. Finishing Installation – Restart!



Installation has finished. You can continue testing Ubuntu now, but until you restart the computer, any changes you make or documents you save will not be preserved.

Figure A-13: installation complete message

And now congratulations you have Ubuntu 12.04 on your device, you can use the same steps to install other versions of Ubuntu

Appendix B: OpenBTS configurations.

```
OpenBTS> config
Control.GSMTAP.GPRS 0     [default]
Control.GSMTAP.GSM 0     [default]
Control.GSMTAP.TargetIP 127.0.0.1     [default]
Control.LUR.404RejectCause 0x04     [default]
Control.LUR.AttachDetach 1     [default]
Control.LUR.FailMode ACCEPT     [default]
Control.LUR.FailedRegistration.Message Your handset is not
provisioned for this network.      [default]
Control.LUR.FailedRegistration.ShortCode 1000     [default]
Control.LUR.NormalRegistration.Message (disabled)
[default]
Control.LUR.NormalRegistration.ShortCode 0000     [default]
Control.LUR.OpenRegistration .*
Control.LUR.OpenRegistration.Message Welcome to the test
network.  Your IMSI is      [default]
Control.LUR.OpenRegistration.Reject (disabled)     [default]
Control.LUR.OpenRegistration.ShortCode 101     [default]
Control.LUR.QueryClassmark 0     [default]
Control.LUR.QueryIMEI 0     [default]
Control.LUR.RegistrationMessageFrequency FIRST     [default]
Control.LUR.SendTMSIs 0     [default]
Control.LUR.UnprovisionedRejectCause 0x04     [default]
Control.Reporting.PhysStatusTable
/var/run/OpenBTS/ChannelTable.db     [default]
Control.Reporting.StatsTable /var/log/OpenBTSStats.db
[default]
Control.Reporting.TMSITable /var/run/OpenBTS/TMSITable.db
[default]
Control.Reporting.TransactionTable
/var/run/OpenBTS/TransactionTable.db     [default]
Control.SMSCB.Table (disabled)     [default]
Control.TMSITable.MaxAge 576     [default]
Control.VEA 0     [default]
GGSN.DNS (disabled)     [default]
GGSN.Firewall.Enable 1     [default]
GGSN.IP.TossDuplicatePackets 0     [default]
GGSN.MS.IP.Base 192.168.99.1     [default]
GGSN.MS.IP.MaxCount 254     [default]
GGSN.MS.IP.Route (disabled)     [default]
GGSN.ShellScript (disabled)     [default]
GPRS.CellOptions.T3168Code 5     [default]
GPRS.CellOptions.T3192Code 0     [default]
GPRS.ChannelCodingControl.RSSI -40     [default]
GPRS.Channels.Min.C0 2     [default]
GPRS.Channels.Min.CN 0     [default]
GPRS.Enable 0     [default]
GPRS.LocalTLLI.Enable 1     [default]
GPRS.Multislot.Max.Downlink 3     [default]
GPRS.Multislot.Max.Uplink 2     [default]
GPRS.NMO 2     [default]
GPRS.Reassign.Enable 1     [default]
GPRS.TBF.EST 1     [default]
GPRS.TBF.Retry 1     [default]
GSM.CCCH.AGCH.QMax 3     [default]
GSM.CellOptions.RADIO-LINK-TIMEOUT 15     [default]
GSM.CellSelection.CELL-RESELECT-HYSTERESIS 3     [default]
GSM.CellSelection.NCCsPermitted 0     [default]
GSM.CellSelection.NECI 1     [default]
GSM.Channels.C1sFirst 0     [default]
GSM.Channels.NumC1s 7     [default]
GSM.Channels.NumC7s 0     [default]
GSM.Channels.SDCCHReserve 0     [default]
GSM.Cipher.CCHBER 0     [default]
GSM.Cipher.Encrypt 0     [default]
GSM.Cipher.RandomNeighbor 0     [default]
GSM.Cipher.ScrambleFiller 0     [default]
GSM.Handover.FailureHoldoff 5     [default]
GSM.Handover.LocalRSSIMin -80     [default]
GSM.Handover.ThresholdDelta 10     [default]
GSM.Identity.BSIC.BCC 2     [default]
GSM.Identity.BSIC.NCC 0     [default]
GSM.Identity.CI 10     [default]
GSM.Identity.LAC 1000     [default]
GSM.Identity.MCC 602
GSM.Identity.MNC 08
GSM.Identity.ShortName Range     [default]
GSM.MS.Power.Damping 75     [default]
GSM.MS.Power.Max 33     [default]
GSM.MS.Power.Min 5     [default]
GSM.MS.TA.Damping 50     [default]
GSM.MS.TA.Max 62     [default]
GSM.MaxSpeechLatency 2     [default]
GSM.Neighbors (disabled)     [default]
GSM.Neighbors.NumToSend 31     [default]
GSM.Ny1 50     [default]
GSM.RACH.AC 0x0400     [default]
GSM.RACH.MaxRetrans 1     [default]
GSM.RACH.TxInteger 14     [default]
GSM.Radio.ARFCNs 1     [default]
GSM.Radio.Band 900     [default]
GSM.Radio.C0 60
GSM.Radio.MaxExpectedDelaySpread 4     [default]
GSM.Radio.PowerManager.MaxAttenDB 10     [default]
GSM.Radio.PowerManager.MinAttenDB 0     [default]
GSM.Radio.RSSITarget -50     [default]
GSM.ShowCountry 0     [default]
GSM.SpeechBuffer 1     [default]
GSM.Timer.Handover.Holdoff 10     [default]
GSM.Timer.T3212 0     [default]
Log.Alarms.Max 20     [default]
Log.Level NOTICE     [default]
Peering.Neighbor.RefreshAge 60     [default]
Peering.NeighborTable.Path /var/run/OpenBTS/NeighborTable.db
[default]
Peering.Port 16001     [default]
```

```
RTP.Range 98     [default]
RTP.Start 16484     [default]
SIP.DTMF.RFC2833 1     [default]
SIP.DTMF.RFC2833.PayloadType 101     [default]
SIP.DTMF.RFC2976 0     [default]
SIP.Local.IP 127.0.0.1     [default]
SIP.Local.Port 5062     [default]
SIP.Proxy.Registration 127.0.0.1:5064     [default]
SIP.Proxy.SMS 127.0.0.1:5063     [default]
SIP.Proxy.Speech 127.0.0.1:5060     [default]
SIP.Proxy.USSD (disabled)     [default]
SIP.RFC3428.NoTrying 0     [default]
SIP.SMSC smsc     [default]
SMS.FakeSrcSMSC 0000     [default]
SMS.MIMEType application/vnd.3gpp.sms     [default]
TRX.Args (disabled)     [default]
TRX.IP 127.0.0.1     [default]
```

Appendix C: Quick Reference

1) Decibels and Decibel Mill watts

Decibels can be used to express ratios between two values or, when paired with a base unit, an absolute value. When written as dB, only a ratio is being expressed. For example, an SNR of 10 dB means that the signal is 10× stronger than the noise. When written as dBm, an absolute value in mill watts is being expressed. For example, GSM handsets can transmit with a maximum of 33 dBm or 2 W.

A convenient factor to keep in mind is that every time a change of 3 dB occurs, the ratio doubles or halves.

Table B-1 dB: Decibels and ratios

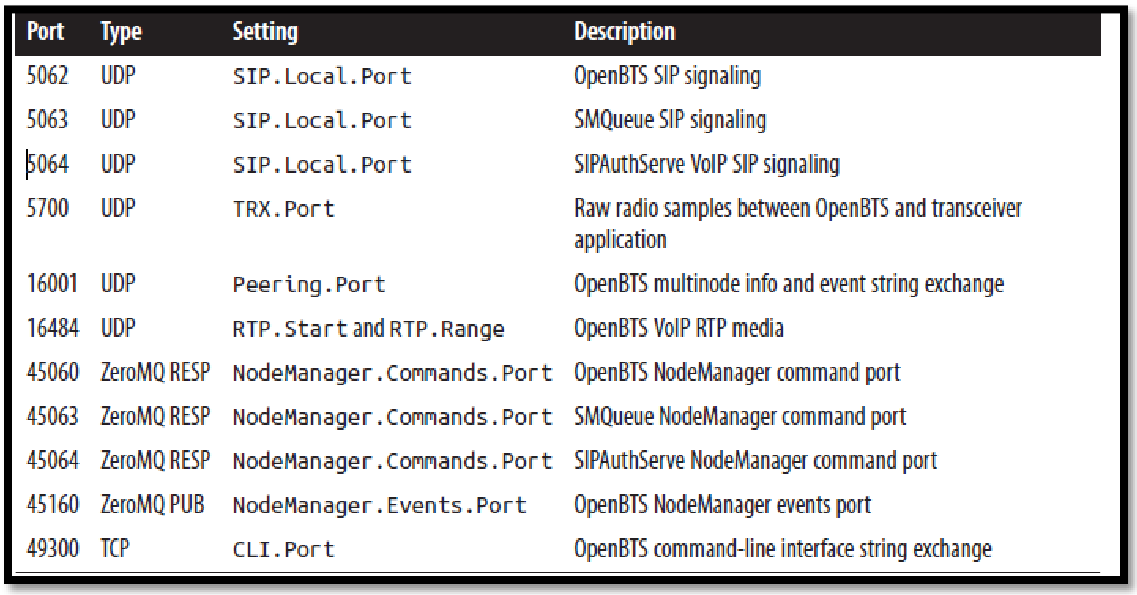
dB	Ratio
30	1000
20	100
10	10
6	~4
3	~2
1	1.259
0	1
-1	0.794
-3	~0.5
-6	~0.25
-10	0.1
-20	0.01
-30	0.001

Table B-2 dBm: Decibel milliwatts

dBm	Absolute value
30	1000 mW or 1 W
20	100 mW
10	10 mW
6	4 mW
3	2 mW
1	1.3 mW
0	1.0 mW or 1000 µW
-1	794 µW
-3	501 µW
-6	251 µW
-10	100 µW
-20	10 µW
-30	1 µW or 1000 nW

2) Network Ports

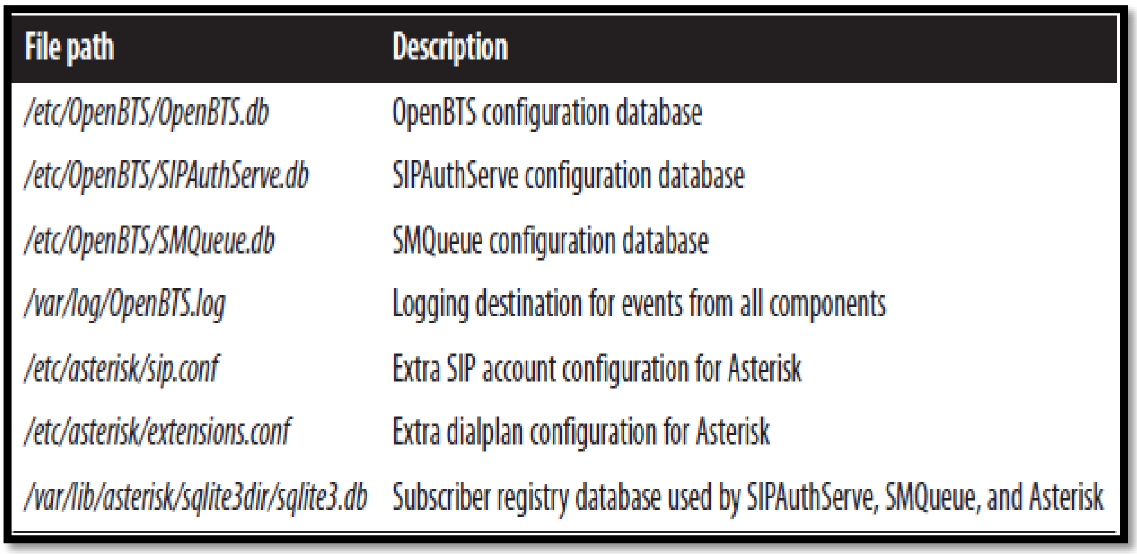
Table B-3 Network Ports



Port	Type	Setting	Description
5062	UDP	`SIP.Local.Port`	OpenBTS SIP signaling
5063	UDP	`SIP.Local.Port`	SMQueue SIP signaling
5064	UDP	`SIP.Local.Port`	SIPAuthServe VoIP SIP signaling
5700	UDP	`TRX.Port`	Raw radio samples between OpenBTS and transceiver application
16001	UDP	`Peering.Port`	OpenBTS multinode info and event string exchange
16484	UDP	`RTP.Start` and `RTP.Range`	OpenBTS VoIP RTP media
45060	ZeroMQ RESP	`NodeManager.Commands.Port`	OpenBTS NodeManager command port
45063	ZeroMQ RESP	`NodeManager.Commands.Port`	SMQueue NodeManager command port
45064	ZeroMQ RESP	`NodeManager.Commands.Port`	SIPAuthServe NodeManager command port
45160	ZeroMQ PUB	`NodeManager.Events.Port`	OpenBTS NodeManager events port
49300	TCP	`CLI.Port`	OpenBTS command-line interface string exchange

3) File Paths

Table B-4File Paths



File path	Description
/etc/OpenBTS/OpenBTS.db	OpenBTS configuration database
/etc/OpenBTS/SIPAuthServe.db	SIPAuthServe configuration database
/etc/OpenBTS/SMQueue.db	SMQueue configuration database
/var/log/OpenBTS.log	Logging destination for events from all components
/etc/asterisk/sip.conf	Extra SIP account configuration for Asterisk
/etc/asterisk/extensions.conf	Extra dialplan configuration for Asterisk
/var/lib/asterisk/sqlite3dir/sqlite3.db	Subscriber registry database used by SIPAuthServe, SMQueue, and Asterisk