

UNIVERSIDADE FEDERAL DE UBERLÂNDIA
FACULDADE DE ENGENHARIA ELÉTRICA
ENGENHARIA ELETRÔNICA E DE TELECOMUNICAÇÕES
CAMPUS PATOS DE MINAS

JULIANA RESENDE

**IMPLEMENTAÇÃO DE UMA ESTAÇÃO RÁDIO BASE UTILIZANDO RÁDIO
DEFINIDO POR *SOFTWARE***

PATOS DE MINAS – MG

2018

JULIANA RESENDE

**IMPLEMENTAÇÃO DE UMA ESTAÇÃO RÁDIO BASE UTILIZANDO RÁDIO
DEFINIDO POR *SOFTWARE***

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica e de Telecomunicações.

Orientador: Prof. Me. Gustavo Nozella Rocha

PATOS DE MINAS – MG

2018

JULIANA RESENDE

**IMPLEMENTAÇÃO DE UMA ESTAÇÃO RÁDIO BASE UTILIZANDO RÁDIO
DEFINIDO POR *SOFTWARE***

Trabalho de Conclusão de Curso apresentado à Faculdade de Engenharia Elétrica da Universidade Federal de Uberlândia, como requisito parcial para obtenção do título de Bacharel em Engenharia Eletrônica e de Telecomunicações.

Patos de Minas, 05 de julho de 2018

Comissão Examinadora

Prof. Me. Gustavo Nozella Rocha (UFU)
Orientador

Prof. Dr. Diego de Brito Piau (UFU)
Examinador

Prof. Me. Júlio César Coelho (UFU)
Examinador

AGRADECIMENTOS

Agradeço primeiramente a Deus por guiar a minha vida e por ter me concedido a oportunidade de estudar e realizar o sonho de ser Engenheira de Eletrônica e de Telecomunicações.

Agradeço aos meus pais Acrisio e Oneida que fazem tudo o que podem para me ver bem e feliz.

Agradeço ao meu irmão Ailton por todo apoio.

Agradeço ao meu namorado Hitalo por todo o carinho, paciência e compreensão.

Agradeço ao meu orientador Prof. Me. Gustavo Nozella Rocha por toda paciência, comprometimento, incentivo e conhecimento compartilhado.

Agradeço a todos que foram meus professores na graduação pelos ensinamentos ao longo desses anos.

RESUMO

Nesse trabalho é mostrado todos os detalhes da implementação de uma Estação Rádio Base GSM utilizando as tecnologias de Rádio Definido por *Software* e *OpenBTS*. É realizado um estudo do Sistema de Telefone Celular, do Sistema Global para Comunicações Móveis (GSM) e das tecnologias envolvidas na implementação dessa Estação Rádio Base (ERB). Diversos temas da área de Telecomunicações voltados para comunicação móvel são abordados no decorrer do trabalho. É apresentada uma descrição do equipamento de Rádio Definido por *Software* e do computador empregados na implementação dessa ERB. Os serviços de chamadas de voz e envio de mensagens SMS utilizando essa ERB foram alcançados com sucesso e são apresentados diversos testes envolvendo tais serviços. Além disso, esse trabalho traz uma abordagem dos benefícios em utilizar tecnologias voltadas para *software* e mostra algumas aplicações práticas relatadas na literatura que uma rede móvel baseada em Rádio Definido por *Software* e *OpenBTS* pode ter.

Palavras-chave: Estação Rádio Base. Rádio Definido por *Software*. *OpenBTS*. GSM.

ABSTRACT

In this work is shown all the details of the implementation of a GSM Base Station Radio using Software Defined Radio and OpenBTS technologies. A study of the Cellular Telephone System, the Global System for Mobile Communications (GSM) and the technologies involved in the implementation of this Base Station Radio (ERB) is carried out. Several topics in the area of Telecommunications aimed at mobile communication are addressed in the course of the work. A description of the Software Defined Radio equipment and the computer used in the implementation of this ERB is presented. The services of voice calls and sending SMS messages using this ERB have been successfully achieved and several tests involving such services are presented. In addition, this work brings a benefits approach to using software-driven technologies and shows some practical applications reported in the literature that a mobile network based on Software Defined Radio and OpenBTS may have.

Keywords: *Base Station Radio. Software Defined Radio. OpenBTS. GSM.*

LISTA DE ILUSTRAÇÕES

Figura 2.1.1 - Esquema de um Sistema Rádio-Celular.....	18
Figura 2.1.2 - Processos de <i>handoff</i> e <i>roaming</i> em um sistema de telefonia celular.....	22
Figura 2.2.1 - Tecnologia de acesso via rádio FDMA / TDMA utilizada no sistema GSM.	24
Figura 2.2.1.1 - Arquitetura do sistema GSM.	25
Figura 2.3.1 - Esquema do Rádio Definido por <i>Software</i> ideal.....	32
Figura 2.3.2 - Esquema do Rádio Definido por <i>Software</i> real.	33
Figura 2.4.1 - Arquitetura de um PBX comum versus a arquitetura do <i>Asterisk</i>	34
Figura 2.5.1 - Arquitetura híbrida do <i>OpenBTS</i>	35
Figura 2.5.2 - Arquitetura final do Sistema de Rede Móvel.	35
Figura 2.5.3 - Cartão inteligente de tamanho real, cartão SIM e cartão Micro SIM.	38
Figura 3.1.1- Rádio Definido por <i>Software</i> USRP N210.	40
Figura 3.2.1 - Bandas GSM disponíveis.....	42
Figura 3.2.2 - Chaves de configuração da categoria <i>GSM.Radio</i>	42
Figura 3.2.3 - Ilustração de um ARFCN.	43
Figura 3.2.4 - Frequências de <i>downlink</i> e <i>uplink</i> para o ARFCN #45 até o ARFCN #55.....	44
Figura 3.2.5 - Canais disponíveis	45
Figura 3.2.6 - Chaves para configuração de canais	45
Figura 3.2.7 - Alterando a chave <i>GSM.Radio.RxGain</i>	46
Figura 3.2.8 - Uso do comando <i>rxgain</i>	46
Figura 3.2.9 - Rede de teste no Moto G.....	47
Figura 3.2.10 - Rede de teste no <i>Xperia E1 D2 114</i>	47
Figura 3.2.11 - Ruído presente no <i>uplink</i>	48
Figura 3.2.12 - USRP N210 com as antenas paralelas.	49
Figura 3.2.13 - USRP N210 com as antenas dispostas formando um ângulo de mais de 90° entre si.	49
Figura 3.2.14 - Ruído presente no <i>uplink</i> com as antenas alinhadas corretamente.	50
Figura 3.2.15 - Potência de transmissão do <i>downlink</i>	50
Figura 3.2.16 - Diminuindo a potência de transmissão do <i>downlink</i> e.....	50
Figura 3.2.17 - Conectividade de <i>uplink</i> extremamente limitada.	52
Figura 3.2.18 - Configurando chaves <i>GSM.Radio.RSSI</i> <i>Target</i> e <i>GPRS.ChannelCodingControl.RSSI</i>	52
Figura 3.2.19 - Conectividade de <i>uplink</i> impossível.	53
Figura 3.3.1 - Executando o <i>Sipauthserve</i>	54
Figura 3.3.2 - Selecionando a rede.	54
Figura 3.3.3 - Aparelho celular informando	55
Figura 3.3.4 - Ativando as trocas IMSI/TMSI.	55
Figura 3.3.5 - Execução do comando <i>tmsis</i>	56
Figura 3.3.6 - IMEI do aparelho celular.	56
Figura 3.4.1 - Execução do programa <i>nmcli.py</i>	58
Figura 3.4.2 - Adicionando assinante na rede.	59
Figura 3.4.3 - Assinante de IMSI 724340300996638 registrado na rede.	59
Figura 3.4.4 - Assinantes cadastrados na rede.	60
Figura 3.5.1 - Alterando o nome da rede.	61
Figura 3.5.2 - Rede JRTelecom.	61
Figura 3.5.3 - Mensagens de registro padrões do <i>OpenBTS</i>	62
Figura 3.5.4 - Alterando mensagem de falha de registro na rede.	62
Figura 3.5.5 - Adicionando mensagem de registro na rede.	62

Figura 3.5.6 - Mensagem de registro na rede.	63
Figura 3.6.1.1 - Tentativa de ingresso na rede por um aparelho celular não assinante.	63
Figura 3.6.1.2 - Causas de rejeições padrão do <i>OpenBTS</i>	64
Figura 3.6.1.3 - Causas de Rejeição disponíveis para não assinantes.	65
Figura 3.6.1.4 - Causas de Rejeição disponíveis para assinantes.	66
Figura 3.6.1.5 - Aplicando causa de rejeição 0x02 para assinantes e não assinantes.	67
Figura 3.6.2.1 - Comando para que o <i>downlink</i> seja transmitido com a máxima potência.	68
Figura 3.6.2.2 - Valores padrão das chaves <i>MS.Power</i>	68
Figura 3.6.2.3 - Valores padrões para a chave <i>GSM.MS.TA</i>	69
Figura 3.6.2.4 - Configurando parâmetro <i>GSM.MS.TA.Max</i> para 10.	69
Figura 3.6.2.5 - Novo valor configurado para o parâmetro <i>GSM.MS.TA.MAX</i>	70
Figura 3.6.3.1 - Valor padrão da chave <i>GSM.Radio.MaxExpectedDelaySpread</i>	70
Figura 3.6.3.2 - Alterando o valor da chave <i>GSM.Radio.MaxExpectedDelaySpread</i>	70
Figura 3.7.1 - Configurações padrões do <i>OpenBTS</i> para o serviço <i>OpenRegistration</i>	71
Figura 3.7.2 - Ativando o serviço <i>OpenRegistration</i>	72
Figura 3.7.3 - Alterando a mensagem de boas-vindas do serviço <i>OpenRegistration</i>	72
Figura 3.7.4 - Novas configurações para o serviço <i>OpenRegistration</i>	72
Figura 3.8.1 - Expansão da rede no <i>OpenBTS</i>	74
Figura 3.8.2 - Parâmetros de Identidade da Estação Rádio Base.	75
Figura 3.8.3 - Mapa de cores.	76
Figura 4.1.1 - Componente <i>smqueue</i>	78
Figura 4.1.2 - Mensagem escrita.	79
Figura 4.1.3 - Retorno da mensagem.	79
Figura 4.1.4 - Envio e retorno de SMS para o número 411.	80
Figura 4.1.5 - Utilizando comando <i>sendsms</i>	81
Figura 4.1.6 - Mensagem recebida.	81
Figura 4.1.7 - Mensagem enviada.	81
Figura 4.1.8 - Mensagem recebida.	82
Figura 4.2.1 - SMS no aparelho de MSISDN 101010.	83
Figura 4.2.2 - SMS no aparelho de MSISDN 505050.	83
Figura 4.2.3 - SMS no aparelho de MSISDN 303030.	84
Figura 4.2.4 - SMS no aparelho de MSISDN 505050.	84
Figura 4.2.5 - Execução do comando <i>stats SMS</i>	85
Figura 4.2.6 - Processo de envio de um SMS entre dois aparelhos celulares.	86
Figura 4.3.1 - Componente <i>Asterisk</i>	86
Figura 4.3.2 - Chamada para 2602.	88
Figura 4.3.3 - Chamada para 2600.	88
Figura 4.3.4 - Informações das chamadas para as extensões 2602 e 2600 geradas no <i>Asterisk</i>	89
Figura 4.4.1 - Origem da chamada MSISDN 303030.	89
Figura 4.4.2 - Destino da chamada MSISDN 101010.	90
Figura 4.5.1.1 - Valores padrões para chaves da categoria <i>GSM.Radio</i>	92
Figura 4.5.1.2 - Teste para aparelho celular <i>Motorola Moto G</i> com chip de IMSI 724340300996638.	94
Figura 4.5.1.3 - Teste para aparelho celular <i>Sony Xperia E1 D2114</i> com chip de IMSI 724312921020978.	94
Figura 4.5.1.4 - Teste para aparelho celular <i>Sony Xperia E1 D2114</i> com chip de IMSI 724312926827553.	95
Figura 4.5.1.5 - Teste para aparelho celular <i>Samsung Galaxy Gran Prime G530BT</i> com chip de IMSI 724340303817952.	95

Figura 4.5.1.6 - Chaves da categoria GSM.Radio configuradas para melhor desempenho.	98
Figura 4.5.1.7 - Teste para aparelho celular <i>Motorola Moto G</i> com chip de IMSI 724340300996638.	99
Figura 4.5.1.8 - Teste para aparelho celular <i>Sony Xperia E1 D2114</i> com chip de IMSI 724312921020978.	99
Figura 4.5.1.9 - Teste para aparelho celular <i>Sony Xperia E1 D2114</i> com chip de IMSI 724312926827553.	100
Figura 4.5.1.10 - Teste para aparelho celular <i>Samsung Galaxy Gran Prime G530BT</i> com chip de IMSI 724340303817952.	100
Figura 4.5.2.1 - Medindo qualidade do <i>link</i> para chamada entre MSISDN 101010 e MSISDN 404040.	103
Figura 4.5.2.2 - Medindo qualidade do <i>link</i> para chamada entre MSISDN 101010 e MSISDN 505050.	104
Figura 4.5.2.3 - Medindo qualidade do <i>link</i> para chamada entre MSISDN 505050 e MSISDN 404040.	104
Figura 4.6.1 - Execução do comando <i>tmsis</i> quando um usuário ingressa na rede através do <i>OpenRegistration</i>	105
Figura 4.6.2 - Mensagem de boas-vindas do serviço <i>OpenRegistration</i>	106
Figura 4.6.3 - Chamada realizada para o número 101.	107
Figura 4.6.4 - Chamada sendo recebida pelo usuário “Assistência”.	107
Figura 4.6.5 - SMS requisitando cadastro na rede.	108
Figura 4.6.6 - Execução do comando <i>tmsis</i> após o usuário do <i>OpenRegistration</i> ser cadastrado na rede.	108
Figura 4.6.7 - Usuário do <i>OpenRegistration</i> presente na lista de assinantes cadastrados na rede.	109
Figura 4.6.8 - SMSs no aparelho de MSISDN 1111122222.	110
Figura 4.6.9 - SMSs no aparelho de MSISDN 505050.	110
Figura 4.6.10 - MSISDN 505050.	111
Figura 4.7.1 - Frequência do sinal de <i>uplink</i>	112
Figura 4.7.2 - Frequência do sinal de <i>downlink</i>	112
Figura 4.8.1 - Processos em execução antes de realizar chamadas.	113
Figura 4.8.2 - Consumo computacional durante uma chamada.	114
Figura 4.8.3 - Consumo computacional durante duas chamadas simultâneas.	114
Figura 4.8.4 - Consumo computacional durante três chamadas simultâneas.	114
Figura 4.8.5 - Consumo computacional durante quatro chamadas simultâneas.	114
Figura 4.8.6 - Consumo computacional durante uma chamada.	115
Figura 4.8.7 - Consumo computacional durante duas chamadas simultâneas.	115
Figura 4.8.8 - Consumo computacional durante três chamadas simultâneas.	115
Figura 4.8.9 - Consumo computacional durante quatro chamadas simultâneas.	116
Figura 4.8.10 - Gráfico: Análise do consumo computacional da CPU para o <i>Transceiver</i>	116
Figura 4.8.11 - Gráfico: Análise do consumo computacional da CPU para o <i>OpenBTS</i>	117
Figura 4.8.12 - Gráfico: Análise do consumo computacional da CPU para o <i>Asterisk</i>	117
Figura 4.9.1 - Rota <i>/var/lib/asterisk/sqlite3dir/sqlite3.db</i>	118
Figura 4.9.2 - Rota <i>/var/lib/asterisk/sqlite3dir/sqlite3.db</i>	119
Figura 4.9.3 - Rota: <i>/etc/OpenBTS/smqueue.db</i>	120
Figura 4.9.4 - Log para eventos que ocorrem no <i>OpenBTS</i>	126
Figura 4.9.5 - Entrada de log quando aparelho celular conecta-se à rede.	127
Figura 4.9.6 - Entradas de log geradas durante troca de SMSs.	127

Quadro 2.2.1 - Faixa de frequência e o número de canais disponíveis para os sistemas GSM 900 e 1800 MHz.	23
Quadro 3.1.1- Especificações técnicas do <i>notebook</i> utilizado.....	40
Quadro 3.1.2 - Especificações do Rádio Definido por <i>Software</i> USRP N210.....	40
Quadro 3.4.1 - Assinantes cadastrados na rede.	60
Quadro 3.7.1 - Expressões regulares e seus efeitos	71
Quadro 4.5.1.1 - Chaves da categoria GSM.Radio	93
Quadro 4.5.1.2 - Aparelhos celulares utilizados no teste de qualidade do <i>link</i>	93
Quadro 4.5.1.3 - Valores da Relação Sinal-Ruído (SNR).	96
Quadro 4.5.1.4 - Valores da potência de transmissão no <i>uplink</i> (TXPWR).....	96
Quadro 4.5.1.5 - Valores do nível do sinal no <i>downlink</i> (RXLEV_DL).....	97
Quadro 4.5.1.6 - Valores da taxa de erro de bit no <i>downlink</i> (BER_DL).	97
Quadro 4.5.1.7 – Valores da taxa de perda de frame de voz no <i>uplink</i> (FER).....	97
Quadro 4.5.1.8 - Chaves da categoria GSM.Radio configuradas para um desempenho melhor.	98
Quadro 4.5.1.9 - Valores da Relação Sinal-Ruído (SNR).	101
Quadro 4.5.1.10 - Valores da potência de transmissão no <i>uplink</i> (TXPWR).....	101
Quadro 4.5.1.11 - Valores do nível do sinal no <i>downlink</i> (RXLEV_DL).....	102
Quadro 4.5.1.12 - Valores da taxa de erro de bit no <i>downlink</i> (BER_DL).	102
Quadro 4.5.1.13 - Valores da taxa de perda de frame de voz no <i>uplink</i> (FER).....	103
Quadro 4.5.2.1 - Testes de Qualidade do <i>link</i> em chamadas entre dois aparelhos celulares..	104
Quadro 4.8.1 - Custo computacional durante a realização de chamadas para a extensão 2602.	116
Quadro 4.9.1 - Registro de SMS.	121
Quadro 4.9.2 - Registro de Chamadas.	122
Quadro 4.9.3 - Registro de Chamadas (Continuação do Quadro 4.9.2).	123
Quadro 4.9.4 - Registro de Chamadas (Continuação do Quadro 4.9.3).	124
Quadro 4.9.5 - Registro de Chamadas (Continuação do Quadro 4.9.4).	125

LISTA DE ABREVIATURAS E SIGLAS

2G - Segunda Geração

3GPP - Projeto de Parceria de Terceira Geração

ADC - Conversor Analógico-Digital

ARFCN - Número Absoluto do Canal de Radiofrequência - *Absolute Radio Frequency Channel Number*

AUC - Centro de Autenticação

BER_DL - Taxa de Erro de *Bit* no *Downlink*

BS - Estação Base

BSC - Controlador da Estação Base

BSS - Subsistema da Estação Base

BTS - Estação Transceptora Base

CAI - *Common Air Interface*

CC - Código do País

CN - *Channel Number*

CPU - Unidade Central de Processamento - *Central Process Unit*

DAC - Conversor Digital-Analógico

DNS - Sistema de Nomes de Domínio - *Domain Name System*

EIR - Registro de Identidade do Equipamento

ERB - Estação Rádio Base

ESN - número de série eletrônico

FAC - Código Localizador de Montagem

FCC - Canais de Controle Direto

FDMA - Acesso Múltiplo por Divisão de Frequência

FER - Taxa de Perda de *Frame* de Voz no *Uplink*

FVC - Canais Diretos de Voz

GMSC - *Gateway* Dedicado MSC

GSM - Sistema Global para Comunicações Móveis

HLR - Registro de Localização Doméstica

IMEI - Identificação Internacional de Equipamento Móvel

IMSI - Identidade Internacional de Assinante Móvel

IPv4 - Protocolo de Internet versão 4

ISC - Centro Internacional de Comutação

ISDN - Rede Digital de Serviços Integrados - *Integrated Services Digital Network*

JSON - Notação de Objetos *JavaScript*

LA - Área de Localização

LAC - Código de Área Local

LAI - Identidade da Área de Localização

LMSI - Identidade de Assinante Móvel Local

LUR - Solicitação de Atualização de Localização - *Location Update Request*

MCC - Código de País Móvel

ME - Equipamento Móvel

MIN - Número de Telefone

MNC - Código de Rede Móvel

MS - Estação Móvel

MSC - Central de Comutação de Telefonia Móvel

MSIN - Número de Identificação do Assinante Móvel

MSISDN ou Número ISDN- Número de Diretório de Assinante Internacional da Estação Móvel

MSRN - Número de *Roaming* da Estação Móvel

MSS - Subsistema de Comutação Móvel

NDC - Código Nacional de Destino

OMC - Centro de Operação de Manutenção

OMSS - Subsistema de Operação e Manutenção

PBX - Troca Automática de Ramais Privados

PIN - Número de Identificação Pessoal

PLMN - Rede Móvel Terrestre Pública

PPA - Arquivos de Pacotes Pessoais - *Personal Package Archives*

PSTN - Rede Pública de Telefonia Comutada

PUK - Chave de Bloqueio Pessoal

QoS - Qualidade de Serviço

RAM - Memória de Acesso Aleatório - *Random Access Memory*

RCC - Canais de Controle Reverso

RSSI - Indicação da Potência do Sinal Recebido - *Received Signal Strength Indication*

RTP - Protocolo de Transporte em Tempo Real

RVC - Canais Reversos de Voz

RXLEV_DL - Nível do Sinal no *Downlink*

SCM - Marca da Classe da Estação

SDCCH - Canal de Controle Dedicado Independente - *Standalone Dedicated Control Channel*

SDR - Rádio Definido por *Software*

SGBD - Sistema de Gerenciamento de Banco de Dados

SIM - Módulo de Identidade do Assinante

SIP - Protocolo de Iniciação de Sessão

SMS - Serviço de Mensagens Curtas - *Short Message Service*

SN - Número de Assinante

SNR - Número de Série - *Serial Number*

SNR - Relação Sinal-Ruído

SP - *Spare*

TA - *Timing Advance*

TAC - Código de Aprovação de Tipo

TCH - Canal de Tráfego - *Traffic Channel*

TDMA - Acesso Múltiplo por Divisão do Tempo

TMSI - Identidade Temporária de Assinante Móvel - *Temporary Mobile Subscriber Identity*

TN – *Timeslot Number*

TXPWR - Potência de Transmissão no *Uplink*

UHD - *Universal Hardware Driver*

USB - *Universal Serial Bus*

USRP - *Universal Software Radio Peripheral*

VLR - Registro de Localização do Visitante

VoIP - Voz sobre IP

SUMÁRIO

1 INTRODUÇÃO.....	16
1.1 Objetivo Geral.....	17
1.2 Objetivo Específico.....	17
2. FUNDAMENTAÇÃO TEÓRICA	18
2.1 Sistemas de Telefonia Celular	18
2.2 Sistema Global para Comunicações Móveis - GSM	23
2.2.1 Arquitetura do Sistema GSM	25
2.2.2 Endereços e Identificadores.....	27
2.2.3 Origem e Encerramento de Chamadas Móveis	30
2.2.4 As Limitações da Rede 2G	31
2.3 Rádio Definido por <i>Software</i>	31
2.4 <i>Asterisk</i>	33
2.5 <i>OpenBTS</i>	34
2.6 Aplicações Encontradas na Literatura	38
3 MATERIAIS E MÉTODOS.....	40
3.1 Procedimentos Iniciais	40
3.2 Configurações Básicas do <i>OpenBTS</i>	41
3.3 Interações entre os Aparelhos Celulares e a Rede	53
3.4 Adicionando Assinantes na Rede	57
3.5 Personalizando a Rede	61
3.6 Ajustes para Otimização do Desempenho da Rede	63
3.6.1 Causas de Rejeição de Assinantes.....	63
3.6.2 Área de Cobertura	67
3.6.3 Distorção do Sinal	70
3.7 Serviço <i>OpenRegistration</i>	71
3.8 Expansão da Rede	73

4 RESULTADOS E DISCUSSÃO	78
4.1 Teste de SMS	78
4.2 Teste de SMS entre dois aparelhos	82
4.3 Teste de Chamadas	86
4.4 Teste de Chamadas Entre Dois Aparelhos Celulares.....	89
4.5 Qualidade do <i>link</i>	90
4.5.1 Qualidade do <i>link</i> em chamadas para a extensão <i>Test Tone Call</i> (2602).....	92
4.5.2 Qualidade do <i>link</i> em chamadas entre dois aparelhos celulares.....	103
4.6 Teste do Serviço <i>OpenRegistration</i>	105
4.7 Verificação das Frequências da Portadora	111
4.8 Análise do Custo Computacional	113
4.9 Armazenamento e Gerenciamento de Dados no <i>OpenBTS</i>	118
5 CONCLUSÃO	128
REFERÊNCIAS	129
APÊNDICE A – Instalação do <i>OpenBTS</i> e seus componentes	132
APÊNDICE B – Conexão do <i>host</i> e do aplicativo <i>transceiver</i> com o USRP N210.....	138
APÊNDICE C – Comandos para inicialização do software <i>OpenBTS</i> e seus componentes..	144
APÊNDICE D – Código para Gráficos	145

1 INTRODUÇÃO

O setor de comunicações móveis por radiofrequência tem evoluído bastante nos últimos anos. Apesar do espectro de frequência ser limitado, a tecnologia dos sistemas de rádio-celular permitiu acomodar muitos usuários em uma extensa área geográfica [1].

No passado, eram os seres humanos que roteavam e ligavam as chamadas, posteriormente as máquinas analógicas, e hoje, os computadores digitais fazem isso automaticamente [2].

Ao longo do tempo, as redes móveis têm sido cada vez mais implantadas e aperfeiçoadas. Entretanto ainda há lugares na Terra desprovidos de linhas telefônicas domésticas ou de recepção de rede móvel. Contudo a maioria desses lugares têm conexão com a Internet via Satélite [2].

Qualquer pessoa que tenha uma conectividade IP pode implementar uma rede móvel através do Projeto *OpenBTS*. Assim, é possível trazer a conectividade para as regiões remotas. Para isso, um telefone com a tecnologia 2G pode se conectar à essa rede móvel e então disponibilizar os serviços de voz ou de SMS [2].

Através da combinação do *OpenBTS* com um equipamento de Rádio Definido por *Software* é possível que a complexa rede móvel seja construída em *software* [2].

Tendo em vista que o Rádio Definido por *Software* é uma tecnologia economicamente viável, multifuncional, programável e de fácil atualização [3], uma rede móvel construída em *software* estará aberta à inovação e seus recursos poderão ser aprimorados com apenas atualizações de *software* [2].

Nos dias atuais, vários projetos de estações rádio base utilizam a arquitetura SDR ou alguma tecnologia baseada nos princípios do SDR [3]. Nesse trabalho é abordado o desenvolvimento de uma Estação Rádio Base GSM (2G) utilizando como base o *OpenBTS* e o Rádio Definido por *Software*.

1.1 Objetivo Geral

Utilizar o projeto *OpenBTS* e um equipamento de Rádio Definido por *Software* para implementar uma Estação Rádio Base GSM.

1.2 Objetivo Específico

Usufruir das inúmeras vantagens proporcionadas pela combinação do Rádio Definido por *Software* com *OpenBTS* para implementar uma Estação Rádio Base GSM. Os aparelhos celulares que estiverem registrados na rede móvel GSM implementada deverão ser capazes de rotear chamadas e enviar SMS.

2. FUNDAMENTAÇÃO TEÓRICA

2.1 Sistemas de Telefonia Celular

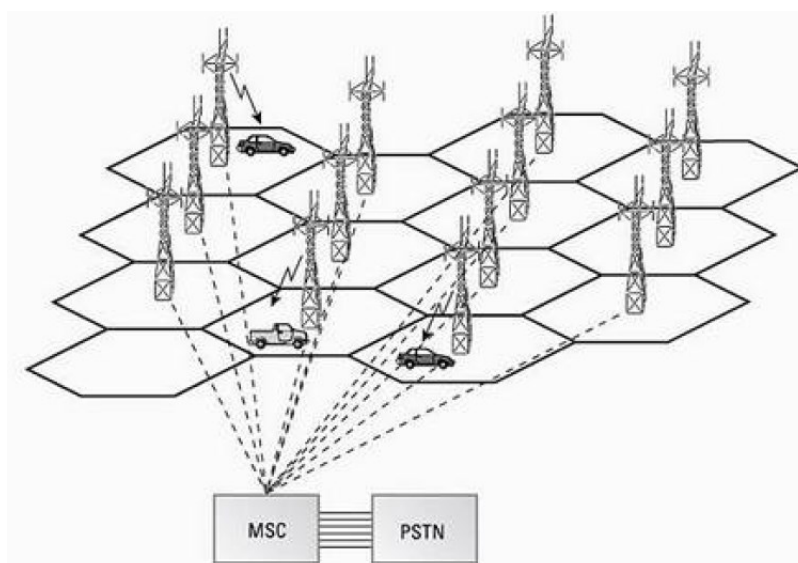
Os sistemas de rádio-celular permitem acomodar muitos usuários em uma extensa área geográfica num espectro de frequência limitado. Os usuários desses sistemas podem conectar-se de forma *wireless* à Rede Pública de Telefonia Comutada (PSTN) em qualquer local que esteja dentro do alcance de rádio do sistema [1].

A cobertura de cada transmissor da estação rádio base é limitada a uma pequena área geográfica denominada célula, de maneira que os mesmos canais de rádio podem ser reutilizados por outra estação rádio base [1].

Em um sistema de telefonia celular, a área de serviço é dividida em regiões chamadas *Clusters* que utilizam todo o espectro de radiofrequências disponível. Os *Clusters* são divididos em células que utilizam um subgrupo do espectro de radiofrequências. Mesmo com o espectro de frequências limitado, os canais utilizados em uma célula podem ser reutilizados em outras se as células pertencem a *Clusters* diferentes e estejam afastadas para diminuir as interferências [4].

A Figura 2.1.1 apresenta um esquema de um sistema de rádio-celular que é composto pelas estações móveis, as estações rádio base, uma Central de Comutação de Telefonia Móvel (MSC) e uma PSTN.

Figura 2.1.1 - Esquema de um Sistema Rádio-Celular.



Fonte: RAPPAPORT, 2009, p. 9.

Cada estação móvel possui um transceptor, uma antena e circuitos de controle. A estação móvel pode ser usada como unidade de mão portátil ou ser montada no interior de um veículo. A comunicação de uma estação móvel a uma estação rádio base é feita por meio de rádio e no decorrer de uma chamada ela pode ser transferida a outras estações rádio base [1].

As estações rádio base possuem vários transmissores, receptores e torres para alocar as antenas de transmissão e recepção, de forma que a comunicação seja feita em modo duplex, ou seja, os sinais são transmitidos em uma frequência e são recebidos em outra frequência de forma simultânea. A estação rádio base pode ser considerada uma ponte entre todos os usuários móveis da célula, sendo responsável por conectar as chamadas móveis à MSC [1].

A MSC coordena as atividades das estações rádio base e tem por função conectar todo o sistema celular à PSTN. Nos grandes centros urbanos são utilizadas várias MSCs por uma única companhia [1]. De acordo com Rappaport (2009): "Uma MSC típica trata de 100 mil assinantes de celular e de 5 mil conversas simultâneas de uma só vez, além de acomodar todas as funções de cobrança e manutenção do sistema [1]."

Toda a comunicação entre estações rádio base e estações móveis é definida pelo padrão CAI (*Common Air Interface*). Esse padrão especifica quatro diferentes canais [1]. São eles:

- 1) Canais Diretos de Voz (FVC): transmitir voz da estação rádio base para as estações móveis;
- 2) Canais Reversos de Voz (RVC): transmitir voz das estações móveis para a estação rádio base;
- 3) Canais de Controle Direto (FCC) e Canais de Controle Reverso (RCC): responsáveis por iniciar as ligações móveis.

Os canais de controle também são chamados de canais de configuração. Eles transmitem e recebem mensagens de dados que transportam solicitações de início de chamada e de serviço, e quando não tem uma chamada em andamento são monitorados pelas estações móveis [1].

Na estação rádio base de uma célula é comum a utilização de 10 a 60 canais de voz e apenas um canal de controle [1].

Os itens, enumerados de 1 a 10, descrevem como é realizada uma chamada a um usuário móvel iniciada por um assinante fixo em um sistema de telefonia celular, fica claro que esses eventos não são observados pelo usuário, pois eles ocorrem em questão de segundos [1]:

- 1) Ao ligar um aparelho de telefone celular, antes de fazer uma chamada, ele varre o grupo de canais de controle direto para determinar um que esteja com o sinal mais forte;
- 2) Esse canal de controle é monitorado até que o sinal caia abaixo de um nível utilizável;
- 3) O aparelho de telefone celular varre os canais de controle para encontrar o sinal de estação rádio base mais forte;
- 4) Quando é feita uma ligação telefônica para um usuário móvel, a MSC envia a solicitação a todas as estações rádio base;
- 5) O número de telefone do assinante que identifica a estação móvel é transmitido como uma mensagem de *paging* para todos os canais de controle direto;
- 6) A estação móvel recebe a mensagem enviada pela estação rádio base e responde identificando-se pelo canal de controle reverso;
- 7) A estação rádio base repassa a confirmação da estação móvel e informa a MSC sobre o *handshake*;
- 8) A MSC instrui a estação rádio base para passar a chamada para um canal de voz livre dentro de uma célula;
- 9) A estação rádio base sinaliza a estação móvel para mudar de frequência para um par de canais de voz direto e reverso não utilizado;
- 10) Uma mensagem de alerta é transmitida pelo canal de voz direto fazendo o telefone móvel tocar e então o usuário móvel atender a chamada.

A conexão de uma chamada iniciada por um usuário móvel em um sistema celular é descrita nos itens enumerados de 1 a 4 [1].

- 1) Uma solicitação de início de chamada é enviada pelo canal de controle reverso quando uma estação móvel origina uma chamada;
- 2) A estação móvel transmite seu número de telefone (MIN), o número de série eletrônico (ESN), o número de telefone da parte chamada e uma Marca da Classe da Estação (SCM) que indica o nível de potência máximo do transmissor para o usuário específico;
- 3) A estação rádio base da célula receberá todos esses dados e os enviará à MSC;

- 4) A MSC valida a solicitação, faz a conexão da parte chamada através da PSTN e instrui a estação rádio base e os usuários do sistema móvel a passar para um par de canais de voz direto e reverso livre, permitindo a conversa.

Embora signifiquem coisas bastante específicas, três temas que causam confusão são a mobilidade, o *handover* e o *roaming* [2]. A seguir é apresentada uma descrição de cada um deles.

A Mobilidade é a capacidade de um aparelho celular receber serviços de diferentes estações rádio base físicas na rede de uma única operadora. A mobilidade só ocorre quando o aparelho celular não está em uma transação ativa, ou seja, chamada de voz ou troca de SMS [2].

À medida que o aparelho celular se move, a qualidade do sinal que este recebe das estações rádio base vizinhas flutuará. Então, quando o aparelho celular detecta um sinal melhor de uma estação rádio base vizinha, ele envia uma LUR (Solicitação de Atualização de Localização - *Location Update Request*) para registrar na nova estação rádio base [2].

Uma LUR na mesma estação rádio base tem por objetivo atualizar um registro já existente. No entanto, quando se tem uma LUR em uma nova estação rádio base, o registro precisa ser alterado para mudá-lo para uma nova torre [2].

O *Handover* é uma técnica em que uma chamada de voz ativa continua ativa à medida que o aparelho celular se move entre estações rádio base [2].

O controlador da estação base (BSC) controla o *Handover* em uma rede GSM tradicional. Entretanto, o *software OpenBTS* elimina a necessidade de um BSC através do uso de um novo protocolo *peer-to-peer*. Assim, informações sobre frequências vizinhas, identidades e chamadas ativas são trocadas por esse protocolo, simplificando a arquitetura de implantação. O aparelho celular obedece aos comandos de *handover* enviados pela rede [2].

Alguns fatores são determinantes na decisão de executar o *handover*. São eles:

- 1) Sinal de *downlink* da estação rádio base em que o aparelho celular está atualmente se torna suficientemente fraco;
- 2) Sinal da estação rádio base vizinha é mais forte, ou seja, excede o nível do sinal da estação rádio base atualmente utilizada;
- 3) Estação rádio base vizinha que está com o sinal mais forte não rejeitou recentemente quaisquer *handovers* devido a congestionamentos.

Caso as três condições mencionadas anteriormente forem atendidas, a estação rádio base a qual o aparelho celular se encontra atualmente iniciará um *handover* para a estação rádio base vizinha que se encontra com o sinal mais forte [2].

Para os assinantes de um sistema celular operarem em áreas de serviço diferentes daquela na qual o serviço é assinado existe um serviço chamado *roaming* que é oferecido por todos os sistemas celulares. Nesse serviço, uma estação móvel é registrada como visitante ao entrar em uma área geográfica diferente de sua área de serviço [1].

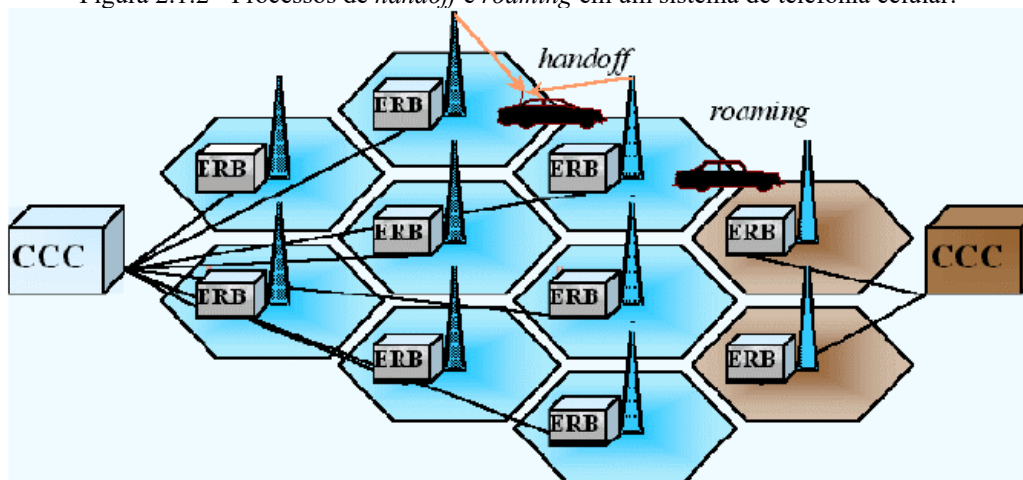
No *roaming* cada visitante está abrigado em um canal de controle direto (FCC) durante todo o tempo. De minutos em minutos a MSC emite um comando global para cada FCC do sistema para pedir a todas as estações móveis que não estavam registradas no sistema informar seu Número de Telefone (MIN) e o Número de Série Eletrônico (ESN) através do canal de controle reverso (RCC) [1].

Assim, novas estações móveis não registradas no sistema transmitem as suas informações de assinante ao receber a solicitação de registro. Essas informações são utilizadas pela MSC para solicitar o status de cobrança do Registro de Localização Doméstica (HLR) a cada estação móvel que está em *roaming* [1].

As estações móveis em *roaming* registradas pela MSC têm permissões para receber e fazer chamadas dessa área, sendo a cobrança roteada de forma automática para o provedor de serviço doméstico do assinante [1].

A Figura 2.1.2 ilustra os processos de *handoff* e *roaming* em um sistema de telefonia celular.

Figura 2.1.2 - Processos de *handoff* e *roaming* em um sistema de telefonia celular.



Fonte: WANDER, 2003, p. 24.

2.2 Sistema Global para Comunicações Móveis - GSM

O Sistema Global para Comunicações Móveis (GSM) é o sistema de telefonia celular de segunda geração (2G) [5]. Ele começou a ser desenvolvido na década de 1980 [6]. Foi através de uma iniciativa europeia com o objetivo de criar um sistema celular móvel uniforme que o padrão GSM foi desenvolvido [5].

Desde 1999, o desenvolvimento e padronização do GSM foi assumido pelo Projeto de Parceria de Terceira Geração (3GPP) que têm como operação o estabelecimento de normas e a publicação de relatórios técnicos sobre a transmissão de dados GPRS (*General Packet Radio Service*) e EDGE (*Enhanced Data Rate for GSM Evolution*) [5].

Os objetivos na concepção do GSM era o desenvolvimento de um sistema digital que permitisse transmissão de voz, mensagens de texto (SMS), transmissão de dados e que fosse um sistema capaz de lidar com o *roaming* internacional [5].

Até o ano de 2005 haviam mais de 1 bilhão de pessoas utilizando o GSM, sua popularidade é devido à recursos como chamadas pré-pagas e *roaming* internacional. Com a tecnologia GSM, os aparelhos passaram a ser menores, mais leves e com mais recursos, não ficando restrito à apenas realizar chamadas. A maior vantagem do GSM foi o aumento da qualidade de voz digital e um baixo custo para a realização de chamadas [6].

Um dos serviços do GSM é o SMS que, por sua vez, foi um sucesso, chegando a quase 15 bilhões de SMS enviados todos os meses até o ano de 2000 [6].

O Quadro 2.2.1 mostra a largura de banda de frequência utilizada para os sistemas GSM de 900 MHz e 1800MHz para o *uplink* (da estação móvel para a estação rádio base) e para o *downlink* (da estação rádio base para a estação móvel) [5].

Quadro 2.2.1 - Faixa de frequência e o número de canais disponíveis para os sistemas GSM 900 e 1800 MHz.

	GSM 900MHz	GSM 1800MHz
<i>Uplink</i> (MHz)	890 - 915	1710 - 1785
<i>Downlink</i> (MHz)	935 - 960	1805 - 1880
Número de canais disponíveis	124	374

Fonte: STASIAK, 2010, p. 4.

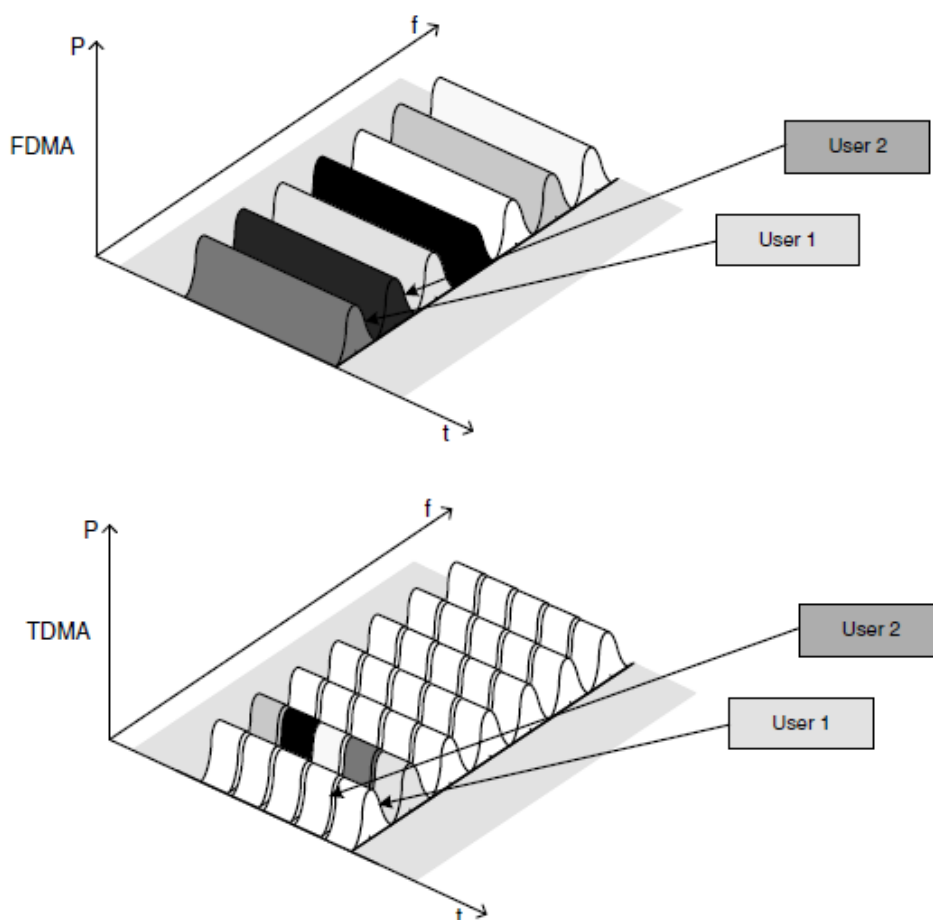
A transmissão duplex para o GSM 900MHz e para o GSM 1800MHz é realizada em intervalos de frequência separados. Cada uma das bandas é dividida em canais com 200KHz de largura de banda, sendo 124 canais disponíveis separadamente para o *uplink* e para a direção

de *downlink* no sistema GSM 900MHz e 374 canais para o GSM 1800MHz [5]. Cada um dos canais de radiofrequência possui 8 canais de fala [6].

De acordo com o tamanho das células, a rede GSM pode ser classificada em macro, micro, pico e guarda-chuva. As células maiores são as macro e as células menores são as pico e as guarda-chuva. O raio das células GSM depende da altura e ganho da antena, condições de propagação, etc., e é por esses fatores que a célula pode ter um alcance de centenas de metros até alguns quilômetros [6].

No sistema GSM, o acesso ao link de rádio é feito utilizando o Acesso Múltiplo por Divisão de Frequência (FDMA) e Acesso Múltiplo por Divisão do Tempo (TDMA) de forma simultânea (Figura 2.2.1). Cada frequência portadora é dividida em oito intervalos de tempo. Para configurar uma conexão é preciso atribuir a cada usuário um canal de frequência e um intervalo de tempo no qual o sinal pode ser transmitido ou recebido [5].

Figura 2.2.1 - Tecnologia de acesso via rádio FDMA / TDMA utilizada no sistema GSM.

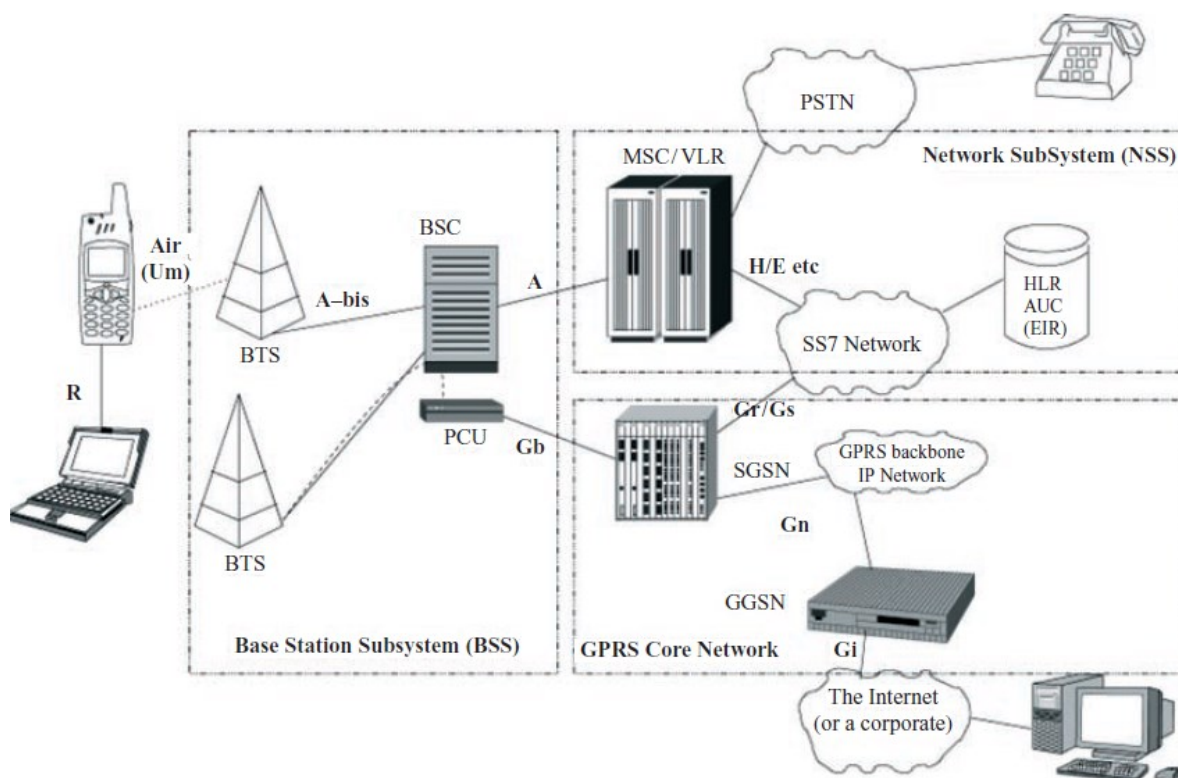


Fonte: STASIAK, 2010, p. 5.

2.2.1 Arquitetura do Sistema GSM

A Figura 2.2.1.1 mostra a arquitetura do sistema GSM.

Figura 2.2.1.1 - Arquitetura do sistema GSM.



Fonte: MISHRA, 2007, p. 9.

Os dois mais importantes componentes da rede de um sistema móvel são a infraestrutura fixa instalada, que é a rede fixa, e os assinantes móveis. A rede fixa é dividida nas sub-redes de rádio, de comutação móvel e de gerenciamento [6].

Para a sub-rede de rádio tem o Subsistema da Estação Base (BSS) que inclui o Controlador da Estação Base (BSC) e a Estação Transceptora Base/Estação Base (BTS/BS). A BTS fica localizada no centro de uma célula, ela é a interface do celular para a rede. O *hardware* do BSC pode estar localizado no mesmo *site* que a BTS, em seu próprio *site* autônomo ou no *site* do Centro de Comutação Móvel [6].

É a BTS que fornece os canais de rádio para sinalização e tráfego de dados do usuário nas células. Ela contém os componentes do transmissor e do receptor que são de alta frequência e também alguns componentes para processamento de sinal e protocolo. Em geral, uma BS contém 1 a 16 transceptores, cada um representando um canal de radiofrequência [6].

São tarefas do BSC: a administração de frequência, o controle da BTS e as funções de troca [6].

Para a sub-rede de Comutação Móvel tem o Subsistema de Comutação Móvel (MSS). O MSS possui centros de comutação móvel e banco de dados que são responsáveis por armazenar os dados necessários para as provisões de roteamento e serviço [6].

O Centro de Comutação Móvel (MSC) é o nó de comutação de uma rede móvel. O MSC é responsável por executar as funções de comutação de um nó de comutação de rede fixa, como por exemplo, realizar a pesquisa de caminho de roteamento e roteamento de sinal [6].

O *Gateway* Dedicado MSC (GMSC) é onde passa o tráfego de voz entre as redes fixas e as redes móveis. Caso a rede fixa não consiga fazer a conexão de uma chamada recebida ao MSC local, ela faz o roteamento da conexão com o GMSC. O GMSC então solicita as informações de roteamento do Registro de Localização Doméstica (HLR) e encaminha a conexão para o MSC local, área onde a estação móvel estará naquele momento [6].

Já as ligações para outras redes móveis internacionais são encaminhadas pelo Centro Internacional de Comutação (ISC) do país correspondente [6].

Para sincronizar o registro de assinantes e a sua localização atual são definidos o Registro de Localização Doméstica (HLR) e o Registro de Localização do Visitante (VLR). Geralmente, há um HLR central por Rede Móvel Terrestre Pública (PLMN) e um VLR para cada MSC [6].

O HLR é responsável por armazenar a identidade e os dados do usuário de todos os assinantes que pertencem à área de um determinado GMSC. Os dados permanentes armazenados são a Identidade de Assinante Móvel Internacional (IMSI) de um usuário em particular, o número de telefone do usuário da rede pública, a chave de autenticação. Os dados temporários do Módulo de Identidade do Assinante (SIM) são o endereço do VLR atual, o número ao qual as chamadas podem ser encaminhadas e parâmetros de trânsito para autenticação e criptografia [6].

O VLR é responsável por armazenar os dados de todas as estações móveis que estão naquele momento na área administrativa do MSC associado. As estações móveis podem estar registradas em um dos VLRs de sua rede doméstica ou em um VLR de uma rede estrangeira, visto que elas podem estar livremente em *roaming*. Cada VLR pode ser responsável por áreas de um ou mais MSCs [6].

Para a sub-rede de Gerenciamento tem o Subsistema de Operação e Manutenção (OMSS) que é responsável por controlar e manter a operação de rede. É através de um Centro

de Operação de Manutenção (OMC) que as funções de controle de rede são monitoradas e iniciadas. O OMC tem acesso ao GMSC e ao BSC. A administração e operações comerciais (assinantes, terminais finais, cobrança e estatística); a gestão de segurança; a configuração de rede, operação, gerenciamento de desempenho e as tarefas de manutenção são todas funções do OMC [6].

Para prover a segurança do sistema, existem dois bancos de dados que baseiam na verificação da identidade do assinante e do equipamento. No Centro de Autenticação (AUC) são armazenados e/ou gerados os dados confidenciais e chaves. No Registro de Identidade do Equipamento (EIR) são armazenados os números de série dos terminais. Esses números de série são fornecidos pelos fabricantes dos equipamentos. É através desse número de série que é possível o bloqueio do acesso de serviço das estações móveis dadas como roubadas [6].

2.2.2 Endereços e Identificadores

A Estação Móvel (MS) compreende todos os equipamentos utilizados pelos assinantes móveis para acessar os serviços. A MS possui dois componentes principais, o Equipamento Móvel (ME) e o Módulo de Identidade do Assinante (SIM) [6].

Além da Identificação Internacional de Equipamento Móvel (IMEI), a estação móvel possui como dados referentes ao assinante o IMSI e o MSISDN (Número de Diretório de Assinante Internacional da Estação Móvel), também chamado de Número ISDN do Assinante Móvel [6].

O Módulo de Identidade do Assinante (SIM) é responsável por fornecer uma identidade ao equipamento móvel. O cartão SIM identifica o assinante na rede e armazena alguns parâmetros do assinante juntamente com os dados pessoais utilizados pelo assinante. Antes de usar o aparelho celular, os assinantes devem digitar o PIN que é um Número de Identificação Pessoal de 4 bits. Caso o PIN seja digitado 3 vezes incorretamente, o cartão é bloqueado e só pode ser desbloqueado com uma Chave de Bloqueio Pessoal de 8 bits (PUK). Tanto o PIN como a PUK são armazenados no cartão SIM [6].

A Identificação Internacional de Equipamento Móvel (IMEI) serve para identificar exclusivamente as estações móveis internacionalmente. O código IMEI é alocado pelo fabricante do equipamento e registrado pelos operadores de rede que o armazena no Registro de Identidade do Equipamento (EIR) [6].

O IMEI, representado pela equação 2.2.2.1, é formado pelo Código de Aprovação de Tipo (TAC), Código Localizador de Montagem (FAC), Número de Série (SNR) e o *spare* (SP). O TAC possui 3 casas decimais que são atribuídas centralmente, o FAC e o SNR possuem 6 casas decimais cada um, ambos atribuídos pelo fabricante e o SP possui uma casa decimal [6].

$$IMEI = TAC + FAC + SNR + SP \quad (2.2.2.1)$$

A Identidade de Assinante Móvel Internacional (IMSI), representada pela equação 2.2.2.2 é um identificador exclusivo que cada assinante recebe ao registrar-se para o serviço com um operador de rede. Um SIM com um IMSI válido deve ser inserido em um equipamento com um IMEI válido para que uma estação móvel possa ser operada. O IMSI é formado pelo Código de País Móvel (MCC), Código de Rede Móvel (MNC) e o Número de Identificação do Assinante Móvel (MSIN). O MCC possui 3 casas decimais que são padronizadas internacionalmente, o MNC possui 2 casas decimais para identificação exclusiva de redes móveis em todo o país e o MSIN possui um máximo de 10 casas decimais para identificação do assinante em sua rede doméstica móvel [6].

$$IMSI = MCC + MNC + MSIN \quad (2.2.2.2)$$

O Número ISDN do Assinante Móvel ou MSISDN, representado pela equação 2.2.2.3 é o número de telefone real da MS. Uma MS, dependendo do SIM, pode ter mais de um MSISDN. Para a seleção de diferentes serviços tais como voz ou dados, um assinante pode armazenar vários MSISDNs, sendo cada MSISDN de um assinante responsável por um determinado serviço. O MSISDN é formado pelo Código do País (CC), o Código Nacional de Destino (NDC) e o Número de Assinante (SN). O CC possui até 3 casas decimais, o NDC possui de 2 a 3 casas decimais e o SN possui no máximo 10 casas decimais [6].

$$MSISDN = CC + NDC + SN \quad (2.2.2.3)$$

O Número de *Roaming* da Estação Móvel (MSRN) é um número ISDN que depende da localização temporária. O MSRN é concedido por um VLR responsável localmente por cada MS em sua área. É através do MSRN que as chamadas são encaminhadas pelo MS. O MSRN é passado do HLR para o GMSC. Ele é formado pelo Código do País da rede visitada (CC), pelo Código Nacional de Destino da rede visitada (NDC) e pelo Número do Assinante (SN) da

rede móvel atual. Tanto o CC como o NDC são determinados de acordo com a rede visitada e dependem da localização temporária. Já o SN é atribuído pelo VLR atual e é único dentro da rede móvel [6].

O MSRN pode ser atribuído pelo VLR em cada registro quando a MS entrar em uma nova Área de Localização (LA). Nessa condição, o MSRN também é transmitido do VLR para o HLR, onde é armazenado para roteamento. Para uma chamada recebida, o MSRN é inicialmente solicitado ao HLR da MS. Assim, o MSC responsável atualmente poderá ser determinado e a chamada ser enviada para este nó de comutação [6].

O MSRN também pode ser atribuído cada vez que o HLR o solicitar para configurar uma conexão para as chamadas recebidas pela estação móvel. Nessa condição, o MSRN não é armazenado no HLR, visto que ele é atribuído somente no momento de configuração das chamadas. Assim, o endereço do VLR atual deve ser armazenado nas tabelas do HLR. Quando a informação de roteamento for solicitada do HLR, o HLR vai para o VLR atual e usa uma identificação exclusiva de assinante (IMSI e MSISDN) para solicitar um MSRN válido [6].

A Identidade da Área de Localização (LAI) é única internacionalmente. Cada LA tem o próprio identificador. Ela é formada pelo Código do País (CC), pelo Código de País Móvel (MNC) e pelo Código de Área Local (LAC). O CC possui 3 casas decimais, o MNC possui 2 casas decimais e o LAC possui no máximo 5 casas decimais [6].

Transmitido pela estação base no Canal de Controle de *Broadcast*, o LAI permite que cada célula seja identificada de forma exclusiva no canal de rádio e cada MS pode determinar a sua localização [6].

Caso a LAI que é ouvida pelo MSC perceba uma mudança de LA, ela solicita a atualização de suas informações de localização no VLR e HLR. Se a conexão para uma chamada recebida for encaminhada para o MSC atual utilizando o MSRN, o LAI é solicitado no VLR, isso faz com que seja determinada a localização precisa da MS [6].

A Identidade Temporária de Assinante Móvel (TMSI) é atribuída pelo VLR que é responsável pela localização atual de um assinante. A TMSI só tem validade na área atendida pelo VLR e é utilizada no lugar do IMSI para a identificação definitiva e endereçamento da MS. Dessa forma, ninguém poderá determinar a identidade do assinante ao ouvir o canal de rádio, já que a TMSI é atribuída durante a presença da MS na área de um VLR. A TMSI é armazenada na MS apenas no VLR, não sendo passada para o HLR [6].

A Identidade de Assinante Móvel Local (LMSI) é uma chave de busca adicional atribuída pelo VLR a cada MS em sua área com o objetivo de acelerar o acesso ao banco de

dados. Toda vez que as mensagens são enviadas para o VLR sobre um MS, o LMSI é adicionado [6].

2.2.3 Origem e Encerramento de Chamadas Móveis

Para melhor entender a origem e o encerramento de chamadas móveis é dado o exemplo: Suponha que uma pessoa faça uma chamada oriunda de um telefone conectado a uma PSTN ou ISDN (Rede Digital de Serviços Integrados - *Integrated Services Digital Network*), a um assinante móvel que vai da cidade A para a cidade B [6].

Com o celular do assinante móvel ligado, a MS busca a rede celular procurando uma faixa de frequência considerável para algum canal de controle transmitido por uma MS próxima. Depois da atualização de localização, a MS acessa a rede e adquire um número de série específico. Após a MS ter registrado sua localização na rede, ela entra em modo que escuta os canais de paginação da BS selecionada [6].

Com o assinante presente na cidade A, a MS identificará uma BS nesta área, porém, à medida que o assinante se desloca para a cidade B, a MS notará que o sinal começará a cair e então, passará a procurar uma outra BS mais favorável [6].

Após a MS identificar uma BS mais favorável, ela examina seus canais de controle para determinar a área de localização à qual pertence. Se a MS se moveu entre BSs em diferentes áreas de localização, então ela faz uma atualização de localização e informa a rede sobre sua nova posição. Entretanto, se ela pertence à mesma área de localização da BS anterior, a MS muda para um canal de paginação na nova BS e continua a monitorar esse canal para chamadas de paginação recebidas [6].

O procedimento de chamada é iniciado pela pessoa que disca o número do assinante móvel. A rede PSTN/ISDN ao receber um número com o código de área, encaminha a chamada para o *switch de gateway* da rede móvel e fornecerá o número de telefone do assinante móvel [6].

O *switch de gateway* da rede móvel faz o HLR da rede móvel recuperar o registro de assinantes. Assim que a chamada chega na MSC, a MS é paginada para avisar sobre a chamada recebida. Então, uma chamada de paginação é emitida por cada BS na área de localização que o assinante está registrado. Após receber a chamada de paginação, a MS responde iniciando o procedimento de acesso que, por sua vez, começa com o envio de uma mensagem da MS para a BS solicitando um canal. A BS responde para a MS, enviando detalhes de um canal dedicado.

A MS retorna esse canal. De modo a garantir a correta identidade do assinante, ocorre um certo grau de *handshaking* [6].

Posteriormente ao canal de sinalização dedicado ser estabelecido, iniciam-se os procedimentos de segurança nesse canal. Logo após, a rede aloca um canal de voz dedicado e a BS e a MS estabelecem uma conexão [6].

Somente após todos os processos descritos acima, que a MS irá começar a tocar e o assinante poderá conversar podendo haver transferências entre diferentes BSs [6].

Ao término da chamada, há uma troca de informações de sinalização que garante que a rede e a MS identificam a conclusão da chamada, então inicia um processo de abertura de novas chamadas. A MS retorna para o modo ocioso e monitora o canal de paginação da célula atual [6].

2.2.4 As Limitações da Rede 2G

Como as redes 2G são projetadas para oferecer serviços de voz aos assinantes, elas possuem uma baixa taxa de transferência que gira em torno de dezenas de *kilobits* por segundo [6].

As redes 2G possuem uma baixa eficiência para serviços de comutação de pacotes, ou seja, o acesso à Internet sem fio com as redes 2G não é implementado com efetividade [6].

Devido a uma infinidade de padrões concorrentes, é permitido a um usuário *roaming* limitado, pois o usuário só pode percorrer as redes que suportam um mesmo padrão [6].

2.3 Rádio Definido por *Software*

Os dispositivos sem fio que são capazes de transmitir ou receber sinais da faixa de radiofrequência do espectro eletromagnético são chamados de Rádio. Um problema em utilizar os dispositivos de Rádio baseados em *hardware* é a necessidade de intervir fisicamente para fazer quaisquer modificações. Porém, esse problema pode ser resolvido com a utilização da tecnologia de Rádio Definido por *Software* (SDR). Essa tecnologia, bastante atraente, permite soluções mais baratas já que os dispositivos permitem modificações através de atualizações de *software*, não havendo necessidade de novos equipamentos [7].

De acordo com o *Wireless Innovation Forum*, Rádio Definido por *Software* é o “Rádio no qual algumas ou todas as funções da camada física são definidas pelo *software*” [7].

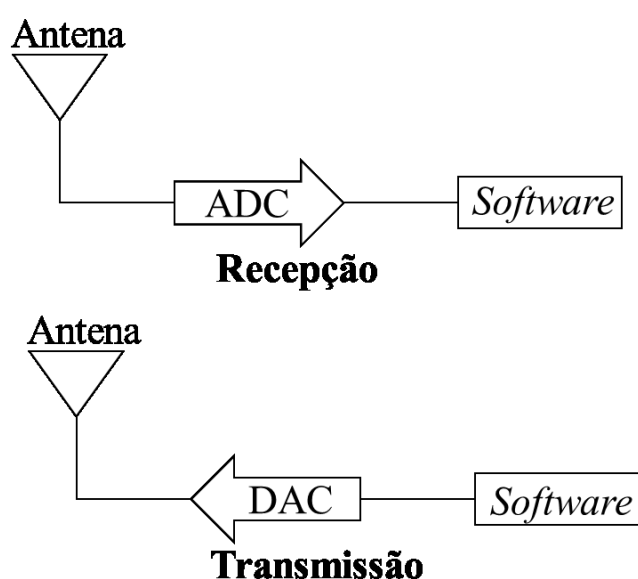
Além de ser economicamente viável, o SDR é um rádio multifuncional, programável e que pode ser atualizado facilmente. Ele permite ser configurado para suportar diversos padrões de formas de onda, bandas de frequência, largura de banda e modos de operação [3].

Os princípios do SDR têm sido cada vez mais adotados no mundo comercial. Nas aplicações de voz, dados e multimídia que exigem vários requisitos de qualidade de serviço (QoS), é ideal a utilização do SDR devido a sua flexibilidade [3].

Nos dias atuais, vários projetos de estações base utilizam a arquitetura SDR ou alguma tecnologia baseada nos princípios do SDR [3].

O Rádio Definido por *Software* ideal seria de acordo com o esquema mostrado na Figura 2.3.1. Esse esquema consiste de uma antena, um bloco chamado *Software* que é onde é feito o processamento do sinal via *software* e um conversor analógico-digital (ADC) para a função de recepção ou um conversor digital-analógico (DAC) para a função de transmissão. Porém, o SDR ideal é altamente inviável de ser realizado na prática para os sinais de alta frequência, pois não há conversores analógico-digital e digital-analógico que trabalhem em altas frequências. Além disso, as antenas são projetadas para faixas de frequências específicas [8].

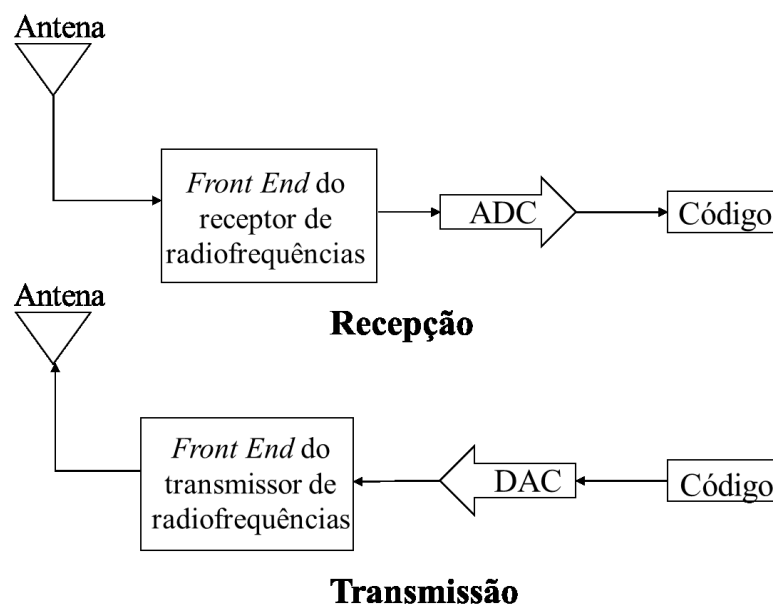
Figura 2.3.1 - Esquema do Rádio Definido por *Software* ideal.



Fonte: ROCHA, 2012, p. 7 (Adaptado).

A alternativa encontrada para obter um SDR real, esquema mostrado na Figura 2.3.2, é utilizar um *Front End* de radiofrequências que é responsável por transladar uma larga faixa do espectro para uma frequência intermediária antes do sinal ser digitalizado [8].

Figura 2.3.2 - Esquema do Rádio Definido por *Software* real.



Fonte: ROCHA, 2012, p. 8 (Adaptado).

2.4 Asterisk

Patrocinado pela *Digium*, o *Asterisk* é uma estrutura gratuita e de código aberto para a construção de aplicativos de comunicação. O *Asterisk* é utilizado em todo o mundo por pequenas e grandes empresas, *call centers*, operadoras e agências que pertencem ao governo. Através do *Asterisk* é possível transformar um computador em um servidor de telecomunicações. O *Asterisk* pode ser utilizado para sistemas IP PBX, *gateways* VoIP, servidores de conferência e outras soluções [9].

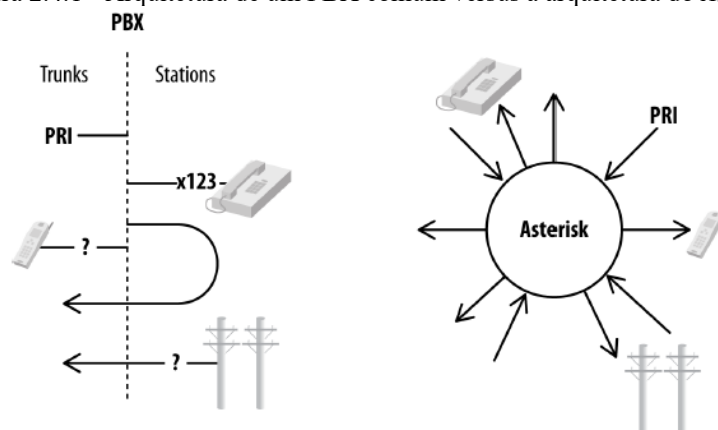
Um PBX (Troca Automática de Ramais Privados) é uma rede de telefonia utilizada por uma empresa que conecta os seus telefones internos à PSTN. Fica claro que o objetivo ao utilizar um PBX não é fornecer serviços de telefonia para o público. O VoIP PBX ou IP PBX é uma das tendências da telefonia PBX que utiliza o protocolo de Internet para transmitir chamadas [10].

O *Asterisk* é bem diferente de outros PBXs tradicionais. Em um tradicional PBX há uma diferença lógica entre as estações que são os telefones e os troncos que são os recursos que

se conectam ao mundo exterior. Nele, não se pode instalar um *gateway* externo em uma porta de estação e rotear chamadas externas para ele sem exigir que os usuários digitem primeiro o número da extensão. Além disso, um recurso fora do *site* é muito mais difícil de ser implementado em um PBX tradicional, pois o sistema não irá permitir que os recursos externos tenham acesso aos recursos internos [11].

No *Asterisk* não há um conceito interno de troncos ou estações. Nele, tudo que entra ou sai do sistema passa por algum canal. A Figura 2.4.1 representa a arquitetura de um PBX comum versus a arquitetura do *Asterisk* [11].

Figura 2.4.1 - Arquitetura de um PBX comum versus a arquitetura do *Asterisk*.



Fonte: MADSEN, 2013, p. 10.

2.5 OpenBTS

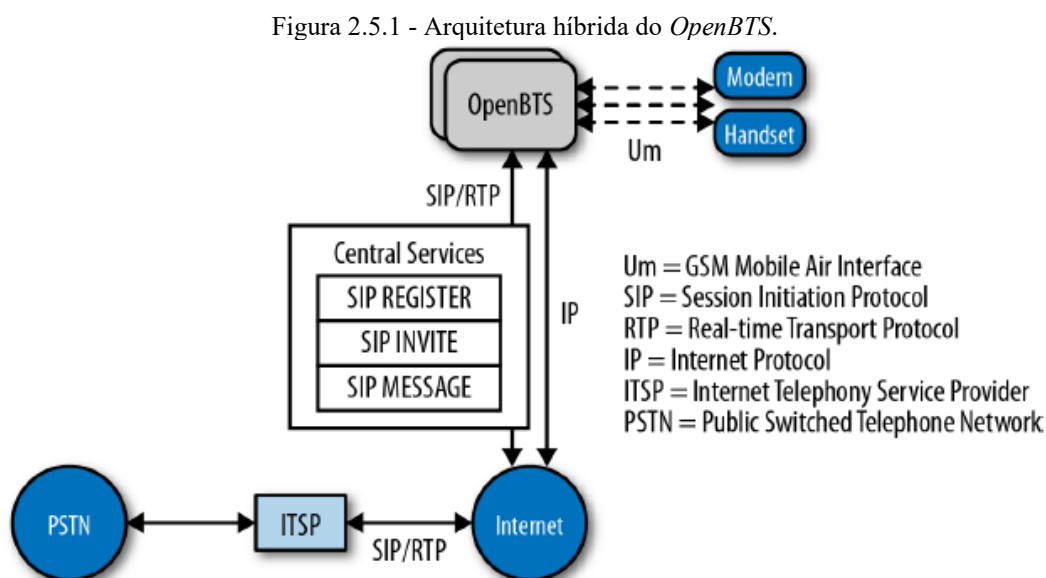
O Projeto de *software OpenBTS* permite que qualquer pessoa que tenha conectividade IP implemente uma rede móvel. Isso é possível devido à conversão entre a interface de rádio sem fio e os protocolos IP abertos. O *OpenBTS* é um aplicativo C++ que implementa a pilha GSM [2].

Embora ainda existam lugares na Terra desprovidos de linhas telefônicas domésticas ou recepção de rede móvel, esses lugares têm uma conexão com a Internet via Satélite. O *OpenBTS* permite converter e distribuir essa conexão à Internet como uma rede móvel em uma extensa região geográfica. Com o *OpenBTS* é possível trazer a conectividade para as regiões remotas, pois qualquer telefone GSM pode se conectar e utilizar os serviços de voz ou de SMS [2].

A combinação de Rádio Definido por *Software* e *OpenBTS* permite a construção de redes de rádio complexas em *software*. Assim, com apenas atualizações de *software* é possível

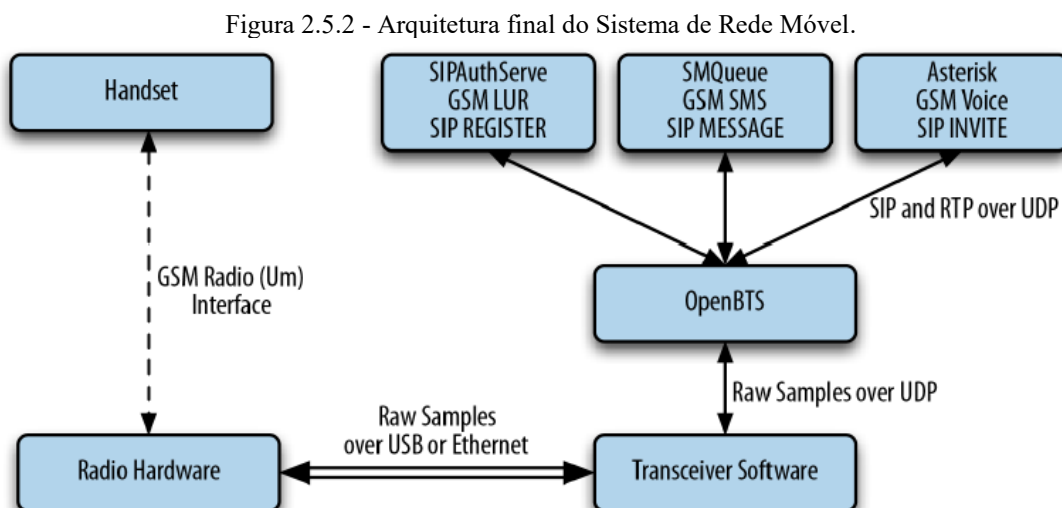
aprimorar os recursos dessas redes. A rede móvel estará aberta à inovação, pois não serão necessárias permissões dos fornecedores de *hardware* para acessar as suas implementações [2].

A Figura 2.5.1 mostra uma arquitetura híbrida em que o *OpenBTS* permite que a interface de rádio da rede móvel tradicional “Um” que é GSM, se conecte diretamente com os protocolos de telefonia da Internet. O núcleo da rede é constituído por protocolos abertos e usa o protocolo IP para transporte [2].



Fonte: IEDEMA, 2015, p. xii.

Com a instalação dos componentes do *OpenBTS*, tudo o que for preciso para os serviços de voz e SMS serão executados em um único sistema como mostra a arquitetura da Figura 2.5.2.



Fonte: IEDEMA, 2015, p. 12.

Para converter o tráfego GSM em VoIP, o *OpenBTS* utiliza o Protocolo de Iniciação de Sessão (SIP) e o Protocolo de Transporte em Tempo Real (RTP) [2].

Para processar os pedidos do *SIP INVITE* e conectar as chamadas é utilizado o comutador VoIP *Asterisk* [2].

Para processar as solicitações do *SIP REGISTER* que o *OpenBTS* gera quando um aparelho tenta se juntar à rede móvel, é utilizado o aplicativo *SIPAuthServe*. Assim, quando um aparelho é autenticado com sucesso, esse aplicativo atualiza o banco de dados de registro de assinantes com o endereço IP da instância do *OpenBTS* que o iniciou, isso permite que outros assinantes liguem para o telefone [2].

O *SIP MESSAGE Queue* ou *SMQueue* é responsável por processar as solicitações do *SIP MESSAGE* geradas pelo *OpenBTS* quando um aparelho celular envia um SMS. Ele tem as funções de armazenar as mensagens, agendar a entrega dessas mensagens na rede e também escalar novamente essas mensagens caso o celular de destino não estiver disponível [2].

O *OpenBTS* é responsável por implementar a Interface Aérea Móvel GSM em *software* para poder se comunicar diretamente com os aparelhos GSM. No lado da rede IP a comunicação é convertida em SIP e RTP e interage com os componentes acima para formar a rede principal. Os aparelhos celulares que têm a tecnologia GSM irão detectar uma rede compatível de rádio GSM [2].

A rede móvel totalmente funcional implementada em *software* aparecerá como qualquer outra rede no aparelho celular. Ela irá rotear chamadas e SMS entre os participantes da rede [2].

Ainda que o *OpenBTS* implemente a maior parte da complexidade da rede móvel em *software*, as ondas de rádio precisam ser transmitidas e recebidas por algum *hardware*. O equipamento de Rádio Definido por *Software* é o *hardware* que tornará possível implementar uma rede móvel em *software* [2].

O equipamento SDR é conectado ao computador por meio de uma interface *USB* ou uma porta *Ethernet*. Esse equipamento implementa um rádio genérico que pode enviar e receber formas de onda na faixa de frequência de 60 MHz a 4 GHz com uma aplicação host. O *OpenBTS* suporta SDRs dos fabricantes *Ettus Research*, *Fairwaves*, *Nuand* e *Range Networks* [2].

Um servidor *Linux* é um requisito para construir essa rede móvel. Este computador pode ser uma máquina instalada em um ambiente de teste ou uma máquina virtual instalada em um *laptop* [2].

Para uma configuração com um único sinal de portadora que permitirá um máximo de sete canais de voz simultâneos, é recomendado no mínimo um processador *Intel i5* e 2 GB de *RAM*. Além disso, é preciso ter pelo menos uma interface *USB2*, embora a *USB3* tem se tornado um requisito dos equipamentos de Rádio Definido por *Software* mais atuais. Essa necessidade de uma interface *USB* com maior rendimento está relacionada com a quantidade e tamanho de amostras de ondas de rádio que são comunicadas por meio dela. Nos ambientes de produção, podem ser utilizados múltiplos sinais portadores ao mesmo tempo, o que gera um aumento da amostra da largura de banda [2].

Para um único sinal de suporte, o *OpenBTS* deve gerar as formas de onda de ligação descendente para transmitir ao aparelho celular e demodular as formas de onda ascendente recebidas do aparelho celular. É possível aumentar a capacidade da rede, pois o *OpenBTS* suporta a criação de múltiplos sinais em um único equipamento de rádio. Entretanto, isso implica em demandas de processamento muito altas [2].

Em geral, uma área de cobertura com raio de 1 metro pode ser alcançada já que muitos SDRs têm sensibilidade de transmissão e recepção para operar sem antenas em um ambiente pequeno. Isso é uma configuração desejável para trabalhos feitos em laboratórios, pois a rede não irá interferir em nenhuma operadora da área [2].

Se adicionar um par de pequenas antenas com ganho de 5 dBi, a área de cobertura poderá ser expandida até um raio de aproximadamente 25 metros em um ambiente desobstruído. Como as antenas são sintonizadas em uma frequência específica, deve ser escolhida uma que corresponda mais à banda GSM utilizada [2].

O tamanho da área de cobertura também sofre influência da frequência. As bandas de baixa frequência propagam distâncias maiores, podendo chegar ao dobro da distância que as bandas de alta frequência alcançariam [2].

Para que a rede móvel funcione é preciso utilizar aparelhos celulares desbloqueados que aceitem o Cartão de Módulo de Identidade de Assinante (SIM). Um aparelho celular bloqueado significa que o processador de banda base do *hardware* foi programado pelo fabricante para trabalhar somente com uma determinada operadora. Entretanto, há meios para remover essa restrição [2].

O cartão SIM utilizado nos aparelhos GSM é também chamado de chip ou de placa de circuito integrado. O SIM é um cartão inteligente que é programado por meio de escritores de cartões inteligentes padrão. Ele é utilizado em aplicações que exigem segurança e autenticação [2].

A Figura 2.5.3 mostra um cartão inteligente de tamanho real que tem as dimensões de um cartão de crédito, o cartão SIM e o cartão Micro SIM.

Figura 2.5.3 - Cartão inteligente de tamanho real, cartão SIM e cartão Micro SIM.



Fonte: IEDEMA, 2015, p. 5.

2.6 Aplicações Encontradas na Literatura

Em [12] é descrita a utilização de *OpenBTS* juntamente com Rádio Definido por *Software* para criar uma BTS (Estação Transceptora Base) para restabelecer a conexão sem fio em áreas que foram atingidas por uma grande catástrofe. Assim, qualquer pessoa que tiver um aparelho celular que funcione na banda GSM poderá ter conectividade sem nenhum custo adicional.

Em [13] é descrita uma maneira de estabelecer um sistema de comunicação temporário de emergência em áreas afetadas por desastres onde a infraestrutura de comunicação foi destruída. Com um sistema de comunicação destruído é muito mais complicado fazer operações de socorro. É utilizado um equipamento de SDR, a plataforma *GNU Radio* e também o *OpenBTS*. Além de haver comunicação direta entre as vítimas e os trabalhadores que prestam socorro, é possível, com um aplicativo pré-instalado no celular das vítimas, receber muitas informações sobre elas. Informações sobre identidade, condição física e localização são enviadas à BS de modo a ajudar o trabalho de resgate.

Em [14] é apresentado um projeto em andamento que tem por objetivo construir uma rede GSM utilizando projetos *open-source* e *open hardware*, entre eles o *OpenBTS* e o *Asterisk*, para levar serviços de telefonia de baixo custo para áreas remotas da região Amazônica, onde

os cidadãos não têm acesso a um telefone nem mesmo para necessidades básicas de comunicação.

Tendo em vista que muitas vítimas em situação de desastres ainda têm acesso ao seu aparelho celular e que uma rede celular baseada em *OpenBTS* é uma solução ideal para isso, os autores de [15] apresentam um método para permitir que os aparelhos celulares de vítimas de situações de desastre se associem de forma automática à rede celular *OpenBTS*. Isso porque o aparelho celular das vítimas só é associado à rede de provedores do cartão SIM, e embora as vítimas pudessem associar manualmente à rede disponível, a situação em que elas se encontram e também o conhecimento delas poderiam impedi-las de realizar isso.

3 MATERIAIS E MÉTODOS

3.1 Procedimentos Iniciais

A distribuição *Linux* utilizada foi a Ubuntu 16.04.4 LTS (*Xenial Xerus*) - *Desktop* de 64 bits. Esse sistema operacional foi instalado em um *notebook* com as especificações técnicas apresentadas no Quadro 3.1.1.

O usuário “openbts” deve estar presente no sistema. Isso é configurado durante a instalação do sistema operacional. Deve digitar “openbts” nos campos “Seu nome” e “Escolha um nome de usuário” [2].

Quadro 3.1.1- Especificações técnicas do *notebook* utilizado.

Modelo	<i>Lenovo ideapad 310 – 14ISK 80UG</i>
Processador	<i>Intel(R) Core(TM) i7-6500U CPU @ 2.50GHz 2.59GHz</i>
Memória RAM	8,00 GB
Conectividade de Rede	<i>Gigabit Ethernet</i>

Fonte: A autora.

O *hardware* de Rádio Definido por *Software* utilizado foi o USRP N210, mostrado na figura 3.1.1 do fabricante *Ettus Research*. O Quadro 3.1.2 apresenta algumas das especificações técnicas desse *hardware*.

Figura 3.1.1- Rádio Definido por *Software* USRP N210.



Fonte: [16].

Quadro 3.1.2 - Especificações do Rádio Definido por *Software* USRP N210.

Características Gerais
50 MHz de largura de banda de RF com amostras de 8 bits
25 MHz de largura de banda de RF com amostras de 16 bits
Interface <i>Gigabit Ethernet</i> para <i>Host</i>
Interface de Expansão de 2 Gbps
Resolução ADC 14 bits
Taxa de amostragem ADC 100 MS / s
Resolução DAC 16 bits
Taxa de amostragem DAC 400 MS / s

Taxa de amostragem <i>Host</i> (8b/16b) 50/25 MS/s
DDC/DUC com resolução de 25 mHz
Referência externa de <i>clock</i> de 5 ou 10 MHz
GPS interno opcional com oscilador de referência bloqueado
Arquitetura modular DC-6 GHz
Versão mínima de UHD requerida: 3.8.0
Driver UHD suporta Linux, Mac OSX, Windows
Energia
Entrada DC de 6 V
Corrente de 1,3 A
<i>Daughterboard</i> 2,3 A
Especificações RF
Ruído de fase 1.8 GHz 10kHz -80 dBc / Hz
Ruído de fase 1.8 GHz 100kHz -100 dBc / Hz
Ruído de fase 1.8 GHz 1MHz -137 dBc / Hz
Potência de saída 15 dBm
Figura de Ruído Típica 5 dB
Especificações Físicas
Dimensões 22 x 16 x 5 cm
Peso 1,2 kg
Faixa de temperatura de operação 0 - 55 ° C

Fontes: [17] e [18] (Adaptado).

Inicialmente, foi realizada toda a configuração do ambiente de desenvolvimento, a instalação do *OpenBTS* e seus componentes (Apêndice A), e os procedimentos para conexão do *host* e do aplicativo *transceiver* (integrado ao *OpenBTS*) com o *hardware* USRP N210 (Apêndice B).

Posteriormente, foi testada a inicialização do *OpenBTS* e cada componente, bem como a confirmação da conectividade entre o *OpenBTS* e o *hardware* USRP N210. O Apêndice C apresenta diversos comandos para inicialização do *OpenBTS* e seus componentes.

3.2 Configurações Básicas do *OpenBTS*

Após confirmar que o *hardware* USRP N210 se comunica satisfatoriamente com o aplicativo *transceiver* do *OpenBTS*, deve-se iniciar o serviço *OpenBTS* através dos seguintes comandos:

```
$ cd /OpenBTS/
$ sudo ./OpenBTS
```

O serviço *OpenBTS* iniciará automaticamente uma instância do *software transceiver* e se conectará ao *hardware* USRP N210. Agora, amostras de rádio são trocadas entre o *software transceiver* e o *software OpenBTS* por um *socket* UDP (*User Datagram Protocol*) local [2].

Para realizar as configurações do *OpenBTS*, em um outro terminal do *Linux*, execute-se o comando a seguir para acessar a interface de linha de comando *OpenBTSCLI*:

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

Posteriormente, na *OpenBTSCLI*, executa-se o comando abaixo para verificar todas as bandas de rádio GSM disponíveis. A execução desse comando é mostrada na Figura 3.2.1.

```
OpenBTS> config GSM.Radio.Band
```

Figura 3.2.1 - Bandas GSM disponíveis.

```
OpenBTS> config GSM.Radio.Band
GSM.Radio.Band 900 [default]
- description: The GSM operating band. Valid values are 850 for GSM850,
900 for PGSM900, 1800 for DCS1800 and 1900 for PCS1900. For non-multiband un
its, this value is dictated by the hardware and should not be changed.
- type: multiple choice
- default value: 900
- visibility level: customer warn - a warning will be presented and confirmat
ion required before changing this sensitive setting
- static: 1
- valid values: 850 = GSM850
- valid values: 900 = PGSM900
- valid values: 1800 = DCS1800
- valid values: 1900 = PCS1900
- scope: value must be the same across all nodes

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

Fonte: A autora.

Como visto na Figura 3.2.1, a banda de rádio deve ser um dos quatro valores de bandas GSM disponíveis em todo o mundo: 850, 900, 1800 ou 1900 MHz.

Posteriormente, execute o comando abaixo para verificar a banda GSM e o ARFCN (Número Absoluto do Canal de Radiofrequência - *Absolute Radio Frequency Channel Number*) em uso. A execução desse comando é mostrada na Figura 3.2.2.

```
OpenBTS> config GSM.Radio
```

Figura 3.2.2 - Chaves de configuração da categoria *GSM.Radio*.

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

Fonte: A autora.

Como pode ser observado na Figura 3.2.2, por padrão, a chave *GSM.Radio.Band* mostra que a banda de 900 MHz está sendo utilizada e a chave *GSM.Radio.C0* indica que o ARFCN #51 nessa banda está selecionado.

Para alterar a banda GSM, por exemplo, para 850 MHz, executa-se o comando a seguir e então reinicia-se o *OpenBTS* para aplicar a mudança, pois trata-se de uma chave estática.

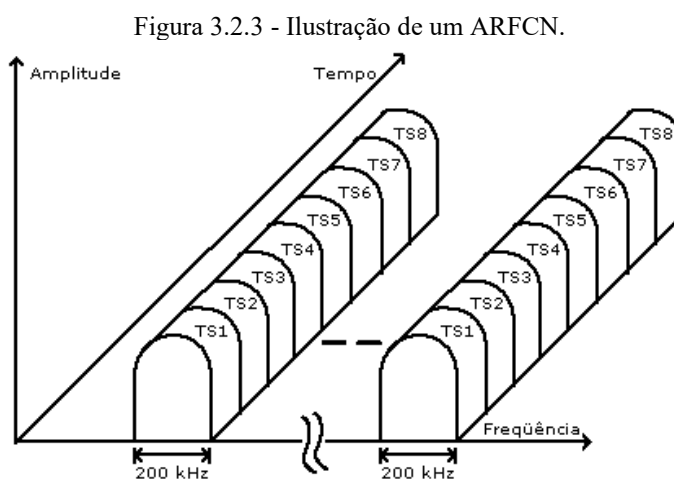
```
OpenBTS> config GSM.Radio.Band 850
```

De acordo com as informações disponíveis no próprio *software OpenBTS*, para a banda de 850 MHz, o intervalo de ARFCN válido é de 128 a 251. Portanto, o ARFCN #51 não é mais válido. Executa-se o comando abaixo para alterar o ARFCN para 150, e novamente, reinicia-se o *OpenBTS*, pois essa também é uma chave estática.

```
OpenBTS> config GSM.Radio.C0 150
```

Entretanto, como o *hardware* de rádio USRP N210 não apresenta limitações para o uso na frequência de 900 MHz, as configurações para essa frequência e o ARFCN #51 foram mantidas. Além disso, utilizando uma frequência mais baixa, obtêm-se melhor cobertura com menos energia [2].

Em uma rede GSM, duas frequências de rádio são utilizadas para que a estação rádio base e os aparelhos celulares possam estabelecer comunicação *full duplex*, ou seja, transmitir e receber dados de forma bidirecional e simultânea. Assim, uma frequência é utilizada para o *downlink* (caminho da estação rádio base para o aparelho celular) e uma outra frequência para o *uplink* (caminho do aparelho celular para a estação rádio base) [1]. A Figura 3.2.3 ilustra um ARFCN, observa-se que são utilizadas duas frequências distintas.



Fonte: [19].

O ARFCN selecionado é o que determina qual par de frequências será utilizado. Um ARFCN é equivalente a uma portadora. Cada banda de rádio tem mais de 100 ARFCNs diferentes [2]. Executa-se o comando a seguir para verificar as frequências utilizadas no *uplink* e no *downlink* para o ARFCN #51. A Figura 3.2.4 mostra uma parte da lista gerada pela execução desse comando onde se encontra o ARFCN de número 51.

```
OpenBTS> config GSM.Radio.C0
```

Figura 3.2.4 - Frequências de *downlink* e *uplink* para o ARFCN #45 até o ARFCN #55.

- valid values:	45 = PGSM900 #45 :	944.201 MHz	downlink /	899.201 MHz	uplink
- valid values:	46 = PGSM900 #46 :	944.401 MHz	downlink /	899.401 MHz	uplink
- valid values:	47 = PGSM900 #47 :	944.601 MHz	downlink /	899.601 MHz	uplink
- valid values:	48 = PGSM900 #48 :	944.801 MHz	downlink /	899.801 MHz	uplink
- valid values:	49 = PGSM900 #49 :	945.001 MHz	downlink /	900.001 MHz	uplink
- valid values:	50 = PGSM900 #50 :	945.201 MHz	downlink /	900.201 MHz	uplink
- valid values:	51 = PGSM900 #51 :	945.401 MHz	downlink /	900.401 MHz	uplink
- valid values:	52 = PGSM900 #52 :	945.601 MHz	downlink /	900.601 MHz	uplink
- valid values:	53 = PGSM900 #53 :	945.801 MHz	downlink /	900.801 MHz	uplink
- valid values:	54 = PGSM900 #54 :	946.001 MHz	downlink /	901.001 MHz	uplink
- valid values:	55 = PGSM900 #55 :	946.201 MHz	downlink /	901.201 MHz	uplink

Fonte: A autora.

Observa-se na Figura 3.2.4 que a frequência para o *downlink* no ARFCN #51 é de 945,401 MHz, e para o ARFCN seguinte (ARFCN #52), a frequência para *downlink* é de 945,601 MHz. Então a largura de banda disponível para o *downlink* no ARFCN #51 é de 200 KHz.

Já para o *uplink*, o ARFCN #51 utiliza a frequência de 900,401 MHz e o ARFCN #52 a frequência de 900,601 MHz. Sendo assim, a largura de banda disponível para o *uplink* também é de 200 KHz.

No sistema GSM cada frequência de portadora é dividida em oito *time slots* [5]. Assim, utilizando um único ARFCN, tem-se oito *timeslots*.

O tráfego de voz e o serviço de GPRS utilizam *timeslots* que transportam canais lógicos TCH (Canal de Tráfego - *Traffic Channel*). Os canais TCH transmitem dados GPRS e voz. Já os canais SDCCH (Canal de Controle Dedicado Independente - *Standalone Dedicated Control Channel*) transportam sinalização, como tráfego de registro do aparelho celular ou tráfego de SMS [2].

Para verificar a quantidade de canais que estão disponíveis para cada tipo de serviço, executa-se o comando a seguir. A execução desse comando é mostrada na Figura 3.2.5.

```
OpenBTS> load
```

Figura 3.2.5 - Canais disponíveis

```

OpenBTS> load
== GSM ==
SDCCH load/available: 0/4
TCH/F load/available: 0/5
PCH load: active, total: 0, 0
AGCH load: active, pending: 0, 0
== GPRS ==
current PDCHs: 2
utilization: 0%

```

Fonte: A autora.

A Figura 3.2.5 mostra que há sete canais TCH disponíveis. Desses sete canais cinco são para o serviço de voz (GSM: TCH/F) e dois para o serviço de dados (GPRS: PDCHs). Esses são valores padrões do *OpenBTS*.

O comando a seguir, permite verificar o valor de várias chaves relacionadas aos canais. A execução desse comando é mostrada na Figura 3.2.6.

```
OpenBTS> config Channels
```

Figura 3.2.6 - Chaves para configuração de canais

```

OpenBTS> config Channels
GPRS.Channels.Min.C0 2      [default]
GPRS.Channels.Min.CN 0      [default]
GSM.Channels.C1sFirst 0     [default]
GSM.Channels.NumC1s auto    [default]
GSM.Channels.NumC7s auto    [default]
GSM.Channels.SDCCHReserve 0 [default]

```

Fonte: A autora.

Embora sejam mantidos os valores padrões para as chaves mostradas na Figura 3.2.6, a chave *GPRS.Channels.Min.C0*, por exemplo, permite configurar o número mínimo de *time slots* TCH disponíveis para serem utilizados no serviço GPRS. Caso a rede seja implantada para atender principalmente o serviço de GPRS ao invés de voz, é possível maximizar a rede para o GPRS alterando o valor dessa chave para sete, ou seja, o número total de canais TCH disponíveis [2]. Para isso, deve ser utilizado o comando a seguir.

```
OpenBTS> config GPRS.Channels.Min.C0 7
```

De acordo com o autor de [2], os equipamentos de rádio do fabricante *Ettus Research* utilizam um valor muito alto para a chave *GSM.Radio.RxGain*, e se esse valor não for ajustado,

o equipamento de rádio não funcionará corretamente, pois o sinal que é recebido sobrecarrega o demodulador. Então, utiliza-se o comando a seguir para alterar o ganho do receptor (*GSM.Radio.RxGain*) de 47 dB para 10 dB (Figura 3.2.7).

```
OpenBTS> devconfig GSM.Radio.RxGain 10
```

Figura 3.2.7 - Alterando a chave *GSM.Radio.RxGain*.

```
OpenBTS> devconfig GSM.Radio.RxGain 10
GSM.Radio.RxGain changed from "47" to "10"
GSM.Radio.RxGain is static; change takes effect on restart
```

Fonte: A autora.

Após o comando anterior, o *OpenBTS* precisa ser reiniciado, pois alterou-se o valor de uma chave que também é estática. No entanto, o comando *rxgain* permite definir o valor do ganho do receptor sem reiniciar o *OpenBTS*. A Figura 3.2.8 mostra o uso desse comando apenas para verificar o ganho do receptor equivalente ao valor da chave *GSM.Radio.RxGain*.

Figura 3.2.8 - Uso do comando *rxgain*.

```
OpenBTS> rxgain
current RX gain is 10 dB
```

Fonte: A autora.

Após os procedimentos anteriores é possível procurar a rede recém-criada em um aparelho celular. Embora o menu de cada aparelho celular seja diferente, basta acessar as configurações de seleção de rede e colocar o aparelho para pesquisar as redes móveis disponíveis. Ao término da pesquisa, haverá uma lista das operadoras de rede disponíveis.

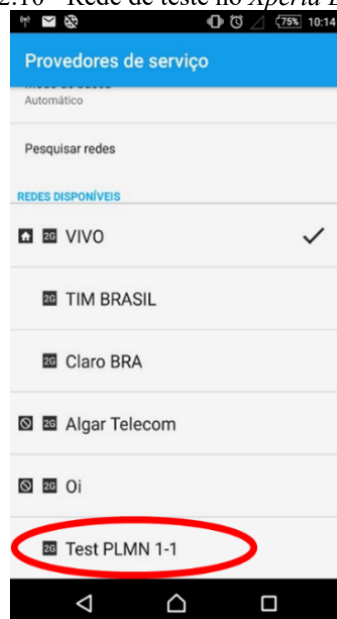
A rede de teste deve aparecer na lista de redes disponíveis com alguns dos seguintes nomes "00101", "001-01", "Teste PLMN 1-1", etc. Esse nome varia de acordo com o modelo do aparelho celular, *firmware* e cartão SIM utilizado [2].

A princípio foram utilizados dois aparelhos celulares para procurar a rede de teste: um da marca *Motorola*, modelo *Moto G*, e outro da marca *Sony*, modelo *Xperia E1 D2 114*. No primeiro, a rede de teste encontrada tem como nome "Test PLMN 001-01" (Figura 3.2.9) e no segundo "Test PLMN 1-1" (Figura 3.2.10).

Figura 3.2.9 - Rede de teste no Moto G.



Fonte: A autora.

Figura 3.2.10 - Rede de teste no *Xperia E1 D2 114*.

Fonte: A autora.

Caso a rede de teste não seja encontrada, o procedimento padrão consiste em forçar a pesquisa de redes novamente, alternar o modo avião entre ligado e desligado ou ainda, desligar e religar o aparelho celular. Além disso, é preciso verificar se o aparelho celular suporta a banda GSM utilizada e se a banda base do aparelho celular está desbloqueada [2].

Conforme evidenciado pelo nome da rede de teste exibido nos aparelhos celulares, fica claro que o *downlink* (caminho da estação rádio base para o aparelho celular) é funcional. Ou seja, o *downlink* da rede está chegando ao aparelho e o sinal está limpo o suficiente para demodular e interpretar as informações [2].

Agora é necessário verificar se o caminho do aparelho celular de volta para a estação rádio base, o *uplink*, é funcional.

Ao configurar uma nova rede é necessário observar o excesso de interferência, ou seja, o ruído gerado por outras fontes no *uplink*. Um *uplink* muito ruidoso significa que os sinais dos aparelhos celulares não podem ser demodulados de forma confiável em informações utilizáveis [2]. O comando abaixo permite verificar o nível de ruído no *uplink* (Figura 3.2.11).

```
OpenBTS> noise
```

Figura 3.2.11 - Ruído presente no *uplink*.

```
OpenBTS> noise
noise RSSI is -79 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

Fonte: A autora.

De acordo com a Figura 3.2.11, a indicação da potência do ruído ambiental detectado, *noise RSSI*, é de -79 dB, isso com as antenas do USRP N210 alinhadas conforme mostra a Figura 3.2.12. Quanto menor o valor de *noise RSSI*, melhor é, pois significa menos ruído presente. Já o parâmetro *MS RSSI target* mostra que a Indicação da Potência do Sinal Recebido (*Received Signal Strength Indication - RSSI*) configurada para os aparelhos celulares é de -50 dB.

Assim, a estação rádio base pode receber 29 dB a mais de energia dos aparelhos celulares do que de ruído ambiental. Isso é uma margem boa e significa que não haverá problemas na conectividade do *uplink* devido ao ruído ambiental.

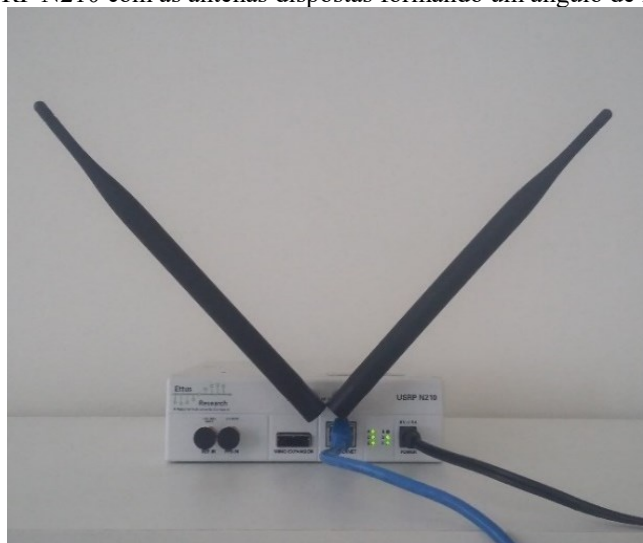
Uma maneira de diminuir o ruído no *uplink* é fazer o correto alinhamento das antenas do *hardware* USRP N210 para que elas não sejam alimentadas umas às outras. Se as antenas estiverem paralelas entre si (Figura 3.2.12), o sinal pode fluir da antena de transmissão para a antena de recepção. No entanto, se as antenas formam um ângulo de 90° ou mais entre si (Figura 3.2.13), o sinal estará sendo transmitido em um plano diferente do que estará sendo recebido [2].

Figura 3.2.12 - USRP N210 com as antenas paralelas.



Fonte: A autora.

Figura 3.2.13 - USRP N210 com as antenas dispostas formando um ângulo de mais de 90° entre si.



Fonte: A autora.

Com as antenas alinhadas conforme mostra a Figura 3.2.13, executou-se novamente o comando *noise* (Figura 3.2.14) e obteve-se uma melhora no nível de ruído do *uplink*. O nível de ruído passou de -79 dB para -83 dB, ou seja, o nível de ruído diminuiu 4 dB apenas com a mudança no alinhamento das antenas. Agora, a estação rádio base pode receber 33 dB a mais de energia dos aparelhos celulares do que ruído ambiental.

Figura 3.2.14 - Ruído presente no *uplink* com as antenas alinhadas corretamente.

```
OpenBTS> noise
noise RSSI is -83 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

Fonte: A autora.

De acordo com o autor de [2], a diminuição da potência de transmissão de *downlink*, limpa ainda mais o sinal de *uplink*. Porém, ao diminuir a potência de transmissão de *downlink* há uma perda da área de cobertura, o que não é significativo em um ambiente de laboratório. Executa-se o comando a seguir para verificar o nível atual da potência de transmissão de *downlink*, que é mostrada na Figura 3.2.15.

```
OpenBTS> power
```

Figura 3.2.15 - Potência de transmissão do *downlink*.

```
OpenBTS> power
current downlink power -10 dB wrt full scale
```

Fonte: A autora.

De acordo com a Figura 3.2.15, o *downlink* está sendo transmitido com menos 10 dB de energia, este é o valor padrão do *OpenBTS*. Para alterar a potência de transmissão do *downlink* basta executar o comando *power* seguido do valor da potência desejada em dB. Executa-se os comandos abaixo para diminuir a potência de transmissão do *downlink* em 30 dB, e em seguida, verificar novamente o nível de ruído no *uplink*. A execução desses comandos é mostrada na Figura 3.2.16.

```
OpenBTS> power 30
OpenBTS> noise
```

Figura 3.2.16 - Diminuindo a potência de transmissão do *downlink* e verificando o nível de ruído no *uplink*.

```
OpenBTS> power 30
current downlink power -30 dB wrt full scale

OpenBTS> noise
noise RSSI is -84 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
INFO: the current noise level is acceptable.
```

Fonte: A autora.

Conforme mostra a Figura 3.2.16 ao atenuar a potência de transmissão do *downlink* em 30 dB, o nível de ruído no *uplink* passou de -83 dB para -84 dB. Isso significa que o nível de ruído no *uplink* diminuiu 1 dB após o aumento da atenuação de 10 dB para 30 dB.

Entretanto, não foi alterado o valor padrão da atenuação da potência de transmissão do *downlink*, uma vez que o sinal de *uplink* já está com uma margem excelente de ruído (-83 dB), evitando assim perda de cobertura. Dessa forma, o *downlink* continuará sendo transmitido com menos 10 dB de energia.

É possível também aumentar a potência de transmissão dos aparelhos celulares. Isso é feito ajustando os valores das chaves *GSM.Radio.RSSITarget* e *GSM.Radio.SNRTarget*, assim os aparelhos celulares recebem instruções para utilizarem mais energia. No entanto, ao fazer isso, os aparelhos celulares descarregarão suas baterias mais rápido, mas os sinais de ligação ascendente serão mais confiáveis. Como os valores padrões para essas chaves são suficientes na maioria das situações, eles não foram alterados [2].

A chave *GSM.Radio.RSSITarget* tem valor padrão de -50 dB, ou seja, o nível RSSI (Indicação da Potência do Sinal Recebido), no *uplink* configurado para aparelhos celulares é de -50 dB. O *loop* de controle de potência do aparelho celular ajusta a potência de transmissão para tentar manter a RSSI no *uplink* a este nível (-50 dB), ou para satisfazer *GSM.Radio.SNRTarget*. Os valores válidos para a chave *GSM.Radio.RSSITarget* são de -75 a -25 dB.

Se o valor da chave *GSM.Radio.RSSITarget* for apenas 10 dB acima do nível de ruído, a conectividade de *uplink* será extremamente limitada, pois o nível de ruído se aproxima da RSSI do aparelho celular [2]. Para exemplificar isso, altera-se o valor da chave *GSM.Radio.RSSITarget* para -75 e então verifica-se o nível de ruído no *uplink*.

Entretanto, ao alterar o valor da chave *GSM.Radio.RSSITarget* para -75, é necessário alterar também a chave *GPRS.ChannelCodingControl.RSSI* para um valor que seja 10 dB acima do valor da chave *GSM.Radio.RSSITarget*. Como o valor da chave *GSM.Radio.RSSITarget* é de -75 dB, alterou-se a chave *GPRS.ChannelCodingControl.RSSI* para -65 dB. A execução dos comandos é mostrada na Figura 3.2.17.

```
OpenBTS> config GSM.Radio.RSSITarget -75
OpenBTS> config GPRS.ChannelCodingControl.RSSI -65
OpenBTS> noise
```

Figura 3.2.17 - Conectividade de *uplink* extremamente limitada.

```
OpenBTS> config GSM.Radio.RSSITarget -75
GSM.Radio.RSSITarget changed from "-50" to "-75"
WARNING: GPRS.ChannelCodingControl.RSSI (-40) should normally be 10db greater than GSM.Radio.RSSITarget (-75)

OpenBTS> config GPRS.ChannelCodingControl.RSSI -65
GPRS.ChannelCodingControl.RSSI changed from "-40" to "-65"

OpenBTS> noise
noise RSSI is -85 dB wrt full scale
MS RSSI target is -75 dB wrt full scale
WARNING: the current noise level is approaching the MS RSSI target, uplink connectivity will be extremely limited.
```

Fonte: A autora.

Conforme mostra a Figura 3.2.17, com o valor da chave *GSM.Radio.RSSITarget* (equivalente ao parâmetro MS RSSI *target*) em 10 dB acima do nível de ruído, a conectividade de *uplink* torna-se extremamente limitada. Então executa-se os comandos a seguir para alterar as chaves *GSM.Radio.RSSITarget* e *GPRS.ChannelCodingControl.RSSI* para os valores padrões, -50 dB e -40 dB respectivamente. A execução desses comandos é mostrada na Figura 3.2.18.

```
OpenBTS> config GSM.Radio.RSSITarget -50
OpenBTS> config GPRS.ChannelCodingControl.RSSI -40
```

Figura 3.2.18 - Configurando chaves *GSM.Radio.RSSITarget* e *GPRS.ChannelCodingControl.RSSI*

```
OpenBTS> config GSM.Radio.RSSITarget -50
GSM.Radio.RSSITarget changed from "-75" to "-50"
WARNING: GPRS.ChannelCodingControl.RSSI (-65) should normally be 10db greater than GSM.Radio.RSSITarget (-50)

OpenBTS> config GPRS.ChannelCodingControl.RSSI -40
GPRS.ChannelCodingControl.RSSI changed from "-65" to "-40"
```

Fonte: A autora.

Caso o valor da chave *GSM.Radio.RSSITarget* (equivalente ao parâmetro MS RSSI *target*) for menor que o nível de ruído, ou seja, o nível de ruído excede a RSSI configurada para o aparelho celular, a conectividade de *uplink* será impossível [2]. Isso é o que acontece quando não altera o ganho do receptor (*GSM.Radio.RxGain*) dos equipamentos de rádio do fabricante *Ettus Research*. Antes de alterar a chave *GSM.Radio.RxGain* de 47 dB para 10 dB, foi executado o comando *noise* e o resultado é o mostrado na Figura 3.2.19.

Figura 3.2.19 - Conectividade de *uplink* impossível.

```
OpenBTS> noise
noise RSSI is -49 dB wrt full scale
MS RSSI target is -50 dB wrt full scale
WARNING: the current noise level exceeds the MS RSSI target, uplink
connectivity will be impossible.
```

Fonte: A autora.

A chave *GSM.Radio.SNRTarget* tem valor padrão igual a 10 dB. O *loop* de controle de potência do aparelho celular ajusta a potência de transmissão para tentar manter a SNR (Relação Sinal-Ruído) acima desse nível ($SNR > 10$). Segundo informações disponíveis no próprio *software OpenBTS*, os valores válidos para essa chave são de 6 a 20.

Após verificar o *uplink* e o *downlink*, é recomendado reiniciar o *OpenBTS* para assegurar-se que todas as configurações sejam aplicadas.

3.3 Interações entre os Aparelhos Celulares e a Rede

É necessário encontrar e inserir os parâmetros de identidade dos aparelhos celulares que serão conectados à rede. O principal parâmetro a ser encontrado é o IMSI (Identidade de Assinante Móvel Internacional). O IMSI, número de 14 a 15 dígitos, armazenado no cartão SIM, é análogo ao nome de usuário do aparelho na rede. Em geral, os aparelhos celulares não divulgam o IMSI de seus SIMs [2].

O *OpenBTS* conhece os IMSIs com os quais interagiu e, estando no controle da rede, é possível ter acesso a essas informações. Para forçar uma interação entre um aparelho celular e a rede de teste, é preciso executar uma operação de Solicitação de Atualização de Localização (LUR - *Location Update Request*) na rede. Isso é feito apenas selecionando a rede de teste na lista de redes disponíveis no aparelho celular [2].

Entretanto, antes de tentar a LUR, é preciso iniciar o *Sipauthserve* (Figura 3.3.1), responsável pelo processamento dessas solicitações. Execute os comandos abaixo em um novo terminal do *Linux* para iniciá-lo:

```
$ cd /OpenBTS/
$ sudo ./sipauthserve
```

Figura 3.3.1 - Executando o *Sipauthserve*.

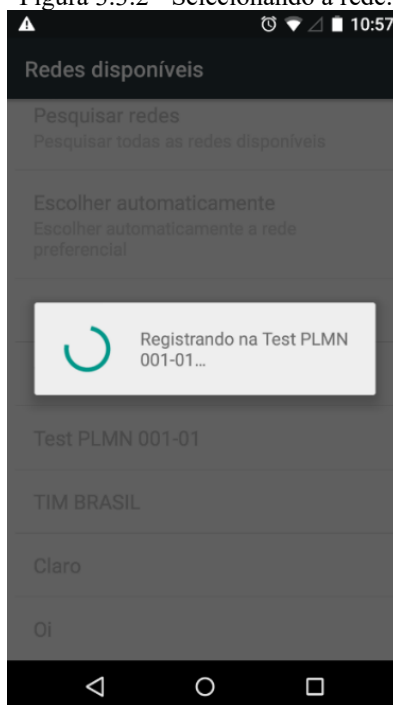
```
openbts@j: /OpenBTS
openbts@j:~$ cd /OpenBTS/
openbts@j:/OpenBTS$ ls
comp128  OpenBTS  OpenBTSCLI  OpenBTSDo  sipauthserve  smqueue  transceiver
openbts@j:/OpenBTS$ sudo ./sipauthserve
[sudo] senha para openbts:
ALERT 9819:9819 2018-05-23T15:47:27.0 sipauthserve.cpp:328:main: ./sipauthserve
(re)starting
```

Fonte: A autora.

Após iniciar o *Sipauthserve*, deve-se selecionar a rede de teste na lista de redes disponíveis no aparelho celular (Figura 3.3.2). Após um curto período de tempo, o aparelho informará uma falha de registro, como mostra a Figura 3.3.3.

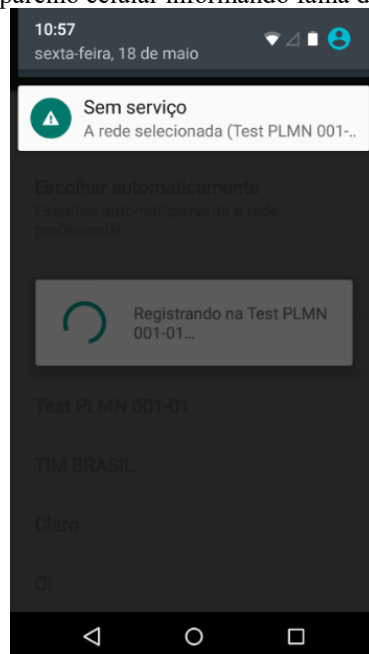
O aparelho celular também pode receber um SMS da rede de teste indicando que o registro falhou. Este SMS inclui automaticamente o IMSI. No entanto, esse recurso não funciona em todos os *hardwares*, como é o caso do aparelho celular aqui utilizado [2].

Figura 3.3.2 - Selecionando a rede.



Fonte: A autora.

Figura 3.3.3 - Aparelho celular informando falha de registro na rede.



Fonte: A autora.

O *OpenBTS* lembra as interações LUR para as trocas IMSI/TMSI (*Temporary Mobile Subscriber Identity*). Isso faz com que o IMSI identificável pelo usuário seja trocado por um TMSI que, por sua vez, é utilizado para aumentar a privacidade do usuário na rede [2].

Por padrão, as trocas IMSI/TMSI estão desativadas. Para ativá-las executa-se o comando abaixo, como mostra a Figura 3.3.4.

```
$ OpenBTS> config Control.LUR.SendTMSIs 1
```

Figura 3.3.4 - Ativando as trocas IMSI/TMSI.

```
OpenBTS> config Control.LUR.SendTMSIs 1
Control.LUR.SendTMSIs changed from "0" to "1"
```

Fonte: A autora.

Agora, para inspecionar todas as interações LUR dos aparelhos celulares com o *OpenBTS*, executa-se o comando a seguir.

```
OpenBTS> tmsis
```

A Figura 3.3.5 mostra uma lista gerada depois da execução do comando *tmsis*. As entradas nessa lista são classificadas por hora, com as primeiras entradas correspondendo às interações mais recentes. Quando há mais de uma interação LUR na rede, é preciso identificar qual aparelho celular corresponde a qual entrada dessa lista. Para isso, pode-se verificar se o

IMEI (Identificação Internacional de Equipamento Móvel) do aparelho celular é o IMEI presente nessa lista. Assim sendo, o IMSI correspondente a esse IMEI, é o IMSI do cartão SIM em uso.

O IMEI é o identificador exclusivo dado ao *hardware* de rádio físico do aparelho celular. Seu valor geralmente é utilizado apenas para relatar e detectar aparelhos celulares roubados. Entretanto, aqui serve como uma maneira conveniente de determinar qual IMSI corresponde a qual IMEI e, conseqüentemente, a qual aparelho celular [2].

Figura 3.3.5 - Execução do comando *tmsis*.



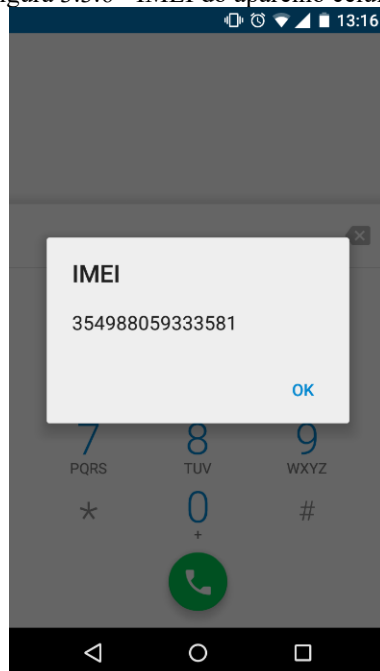
```
OpenBTS> tmsis
```

IMSI	TMSI	IMEI	AUTH	CREATED	ACCESSED	TMSI_ASSIGNED
724340302312523	0x1a389	353954070197700	1	18m	10s	1
724340300996638	0xa7aae	354988059333580	0	76s	76s	0
724340304100051	0xbf161	359059041941720	0	16m	16m	0

Fonte: A autora.

Para descobrir o IMEI de um aparelho celular, digita-se no teclado do aparelho **#06#* ou ainda, pode-se procurar pelo IMEI nas configurações. Para o aparelho celular aqui utilizado, o IMEI é 354988059333581 (Figura 3.3.6).

Figura 3.3.6 - IMEI do aparelho celular.



Fonte: A autora.

De posse do IMEI do aparelho celular, observa-se atentamente na lista da Figura 3.3.5 e verifica-se qual IMSI corresponde ao IMEI 354988059333581. O dígito final do IMEI não

corresponde ao que o *OpenBTS* exibe, pois trata-se de um dígito de verificação que é mostrado como zero no *OpenBTS* [2]. Nesse caso o valor do IMSI correspondente é 724340300996638.

Como pode ser observado na Figura 3.3.5, a coluna AUTH para o aparelho celular de chip com IMSI igual a 724340300996638 se encontra configurada com um zero. Isso é devido ao fato de que a interação LUR falhou, pois não se trata de um assinante conhecido na rede.

Há uma outra entrada na lista com AUTH igual a 1 para o aparelho celular com IMEI igual a 353954070197700. Isso significa que o aparelho celular executou com sucesso a interação LUR, pois ele foi cadastrado com sucesso na rede, ou seja, é um assinante válido.

3.4 Adicionando Assinantes na Rede

De posse dos códigos IMEI e IMSI é possível adicionar um assinante na rede. Porém, ainda é necessário o nome para o assinante, isso para saber qual aparelho ou qual pessoa é o assinante e, também é preciso o MSISDN (Número de Diretório de Assinante Internacional da Estação Móvel), que, nada mais é que o número do telefone do assinante. O MSISDN pode ser qualquer número escolhido, uma vez que não está conectado à rede telefônica pública [2].

Para adicionar assinantes na rede é necessário acessar o programa *nmcli.py*, esse programa permite alterar os parâmetros de configuração, adicionar assinantes e monitorar atividades, tudo isso através de comandos formatados em JSON (Notação de Objetos *JavaScript*). Comandos que são formatados em JSON são de fácil leitura e escrita para os seres humanos e, para as máquinas são fáceis de interpretar e gerar [20].

Para acessar o *nmcli.py*, em um novo terminal do *Linux*, executa-se os seguintes comandos:

```
$ cd dev/NodeManager
$ sudo ./nmcli.py
```

A Figura 3.4.1 mostra a execução do programa *nmcli.py*. Como pode ser observado, há vários comandos para lidar com os assinantes da rede.

Figura 3.4.1 - Execução do programa *nmcli.py*.

```

openbts@j:~/dev/NodeManager$ sudo ./nmcli.py
[sudo] senha para openbts:
targets:
  openbts
  smqueue
  sipauthserve
command examples:
  ./nmcli.py target version                (query the component version)
  ./nmcli.py target config read            (show all configuration values)
  ./nmcli.py target config read Log.Level (show a specific configuration value)
  ./nmcli.py target config update Log.Level INFO (update a specific configuration value)
  ./nmcli.py openbts monitor              (query some live radio data)
  ./nmcli.py sipauthserve subscribers read (show all subscribers)
  ./nmcli.py sipauthserve subscribers create name imsi msisdn (create a subscriber which uses cache auth)
  ./nmcli.py sipauthserve subscribers create name imsi msisdn ki (create a subscriber which uses full auth)
  ./nmcli.py sipauthserve subscribers delete imsi the-imsi (delete a subscriber by imsi)
  ./nmcli.py sipauthserve subscribers delete msisdn the-msisdn (delete a subscriber by imsi)
response codes:
  200 : action ok with response data
  204 : action ok with no response data
  204 : action ok with no change
  404 : unknown key or action
  406 : request is invalid
  409 : conflicting value
  500 : storing new value failed
usage:
  ./nmcli.py target command (action (key (value)))

```

Fonte: A autora.

Utilizando o *nmcli.py* há duas formas para adicionar um assinante na rede. Uma delas cria um assinante que usará a autenticação em cachê e a outra utiliza a autenticação completa. Os comandos abaixo, mostram respectivamente essas duas formas. Esses comandos devem ser executados substituindo “*name*”, “*imsi*”, “*msisdn*” e “*ki*” pelas próprias informações do assinante. O “*imsi*” deve ser substituído pela sigla IMSI seguida pelo valor do IMSI [2].

```

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

```

Como nesse trabalho foram utilizados SIMs de operadoras comerciais, não se tem acesso ao código da chave Ki (Chave de Autenticação do Assinante). Consequentemente, não é possível fazer a autenticação completa.

Para fazer a autenticação completa, é necessário usar SIMs em branco e gravar esses SIMs. Assim o código da chave Ki seria conhecido.

A chave Ki é responsável pela segurança da rede GSM. A conta do assinante é comprometida caso alguém descubra o valor dessa chave e a utilize em um outro cartão SIM. Há diversos tipos de ataques para revelar informações dessa chave. Entretanto, o sistema GSM possui uma função capaz de detectar se dois aparelhos celulares com a mesma identificação estão sendo utilizados ao mesmo tempo, registrando a ocorrência e notificando o usuário [21].

Por questões de segurança, o código da chave Ki é estritamente protegido. Este é armazenado no cartão SIM e no AUC (Centro de Autenticação - *Authentication Center*). O

AUC é responsável pela autenticação dos assinantes na rede. A chave Ki nunca é transmitida pela rede [22].

O comando abaixo adiciona a assinante “Juliana” à rede com número de telefone igual à 101010 e IMSI 724340300996638. A Figura 3.4.2 mostra a execução desse comando.

```
$ ./nmcli.py sipauthserve subscribers create Juliana
IMSI724340300996638 101010
```

Figura 3.4.2 - Adicionando assinante na rede.

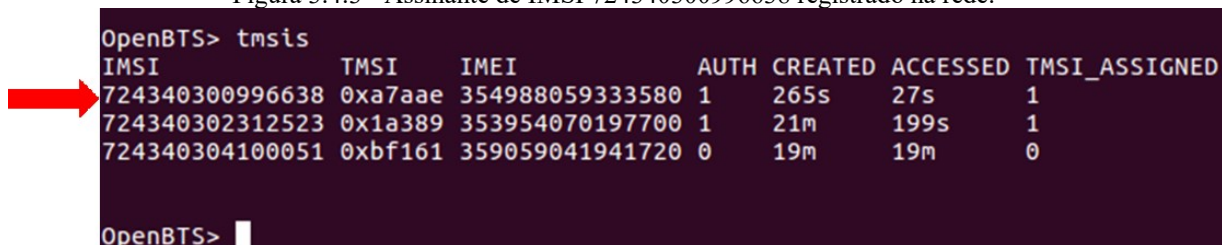
```
openbts@j:~/dev/NodeManager$ ./nmcli.py sipauthserve subscribers create Juliana IMSI
724340300996638 101010
raw request: {"command":"subscribers","action":"create","fields":{"name":"Juliana","
imsi":"IMSI724340300996638","msisdn":"101010","ki":""}}
raw response: {
  "code" : 200,
  "data" : "both ok"
}
```

Fonte: A autora.

Após adicionar a assinante “Juliana” na rede, deve conectar-se à rede de teste na lista de redes disponíveis no aparelho celular. Para assinantes da rede, ela aparece com o nome “Range”. Este nome é a primeira coisa que alguém verá quando procurar pela rede. Entretanto, o nome da rede pode ser alterado, tal procedimento é descrito na seção 3.5.

Posteriormente, executando o comando “*tmsis*”, verifica-se na Figura 3.4.3 que a coluna AUTH para o aparelho celular de chip com IMSI igual a 724340300996638, da assinante “Juliana”, agora se encontra configurada com o número um, ou seja, o aparelho celular executou com sucesso a interação LUR e é um assinante registrado na rede.

Figura 3.4.3 - Assinante de IMSI 724340300996638 registrado na rede.



```
OpenBTS> tmsis
```

IMSI	TMSI	IMEI	AUTH	CREATED	ACCESSED	TMSI_ASSIGNED
724340300996638	0xa7aae	354988059333580	1	265s	27s	1
724340302312523	0x1a389	353954070197700	1	21m	199s	1
724340304100051	0xbf161	359059041941720	0	19m	19m	0

```
OpenBTS> 
```

Fonte: A autora.

No total, foram cadastrados 6 assinantes na rede. O nome dos assinantes, os valores de IMEI, IMSI, MSISDN, a marca e o modelo dos aparelhos celulares utilizados para esses assinantes são mostrados no Quadro 3.4.1.

Quadro 3.4.1 - Assinantes cadastrados na rede.

NOME	IMEI	IMSI	MSISDN	Marca/Modelo Aparelho Celular
Juliana	354988059333581	724340300996638	101010	Motorola Moto G
Gustavo	353954070197703	724340302312523	202020	Samsung Galaxy S6 Edge+
Sony1	322807062050689	724312921020978	303030	Sony Xperia E1 D2114
Sony2	352807062050697	724312926827553	404040	Sony Xperia E1 D2114
Samsung1	359888064927739	724340303817952	505050	Samsung Galaxy Gran Prime G530BT
Assistência	359889064927737	724109393365788	101	Samsung Galaxy Gran Prime G530BT

Fonte: A autora.

Executa-se o comando a seguir para exibir todos os assinantes cadastrados na rede (Figura 3.4.4).

```
$ ./nmcli.py sipauthserve read
```

Figura 3.4.4 - Assinantes cadastrados na rede.

```
openbts@j:~/dev/NodeManager$ ./nmcli.py sipauthserve subscribers read
raw request: {"command":"subscribers","action":"read","key":"","value":""}
raw response: {
  "code" : 200,
  "data" : [
    {
      "imsi" : "IMSI724340300996638",
      "msisdn" : "101010",
      "name" : "Juliana"
    },
    {
      "imsi" : "IMSI724340302312523",
      "msisdn" : "202020",
      "name" : "Gustavo"
    },
    {
      "imsi" : "IMSI724312921020978",
      "msisdn" : "303030",
      "name" : "Sony1"
    },
    {
      "imsi" : "IMSI724312926827553",
      "msisdn" : "404040",
      "name" : "Sony2"
    },
    {
      "imsi" : "IMSI724340303817952",
      "msisdn" : "505050",
      "name" : "Samsung1"
    },
    {
      "imsi" : "IMSI724109393365788",
      "msisdn" : "101",
      "name" : "Assistência"
    }
  ]
}
```

Fonte: A autora.

O comando a seguir permite excluir um assinante da rede pelo IMSI, ou seja, substituindo “the-imsi” por “IMSI” seguido do valor do IMSI:

```
$ ./nmcli.py sipauthserve subscribers delete imsi the-imsi
```

3.5 Personalizando a Rede

O nome padrão da rede que aparece nos aparelhos celulares é “*Range*”, porém é possível alterá-lo, desde que não haja espaço e nem caracteres especiais [2].

O comando a seguir altera o nome da rede para JRTelecom. A Figura 3.5.1 mostra a execução desse comando e a Figura 3.5.2 mostra a rede com o novo nome sendo exibida na lista de redes disponíveis em um aparelho celular.

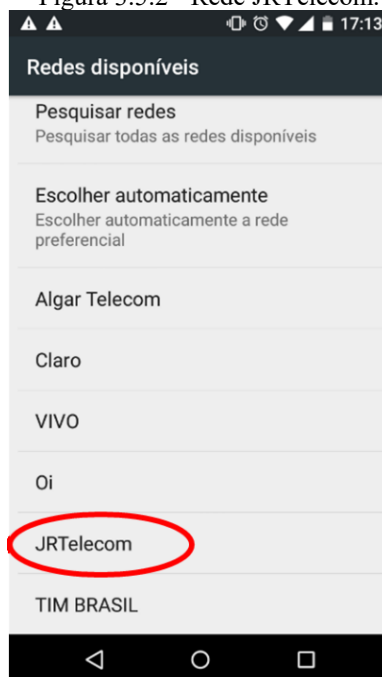
```
OpenBTS> config GSM.Identity.ShortName JRTelecom
```

Figura 3.5.1 - Alterando o nome da rede.

```
OpenBTS> config GSM.Identity.ShortName JRTelecom
GSM.Identity.ShortName changed from "Range" to "JRTelecom"
```

Fonte: A autora.

Figura 3.5.2 - Rede JRTelecom.



Fonte: A autora

Além de alterar o nome da rede, é possível também alterar as mensagens de registro, que são enviadas ao aparelho celular na presença de alguns eventos. Para exibir essas mensagens execute o comando a seguir (Figura 3.5.3).

```
OpenBTS> config Registration.Message
```

Figura 3.5.3 - Mensagens de registro padrões do *OpenBTS*.

```
OpenBTS> config Registration.Message
Control.LUR.FailedRegistration.Message Your handset is not provisioned
for this network. [default]
Control.LUR.NormalRegistration.Message (disabled) [default]
Control.LUR.OpenRegistration.Message Welcome to the test network. Your
IMSI is [default]
```

Fonte: A autora.

Para modificar a mensagem de falha de registro na rede para “Falha de Registro na Rede JRTelecom” (Figura 3.5.4), o comando a seguir é executado.

```
OpenBTS> config Control.LUR.FailedRegistration.Message Falha de
Registro na Rede JRTelecom
```

Figura 3.5.4 - Alterando mensagem de falha de registro na rede.

```
OpenBTS> config Control.LUR.FailedRegistration.Message Falha de Registro
na Rede JRTelecom
Control.LUR.FailedRegistration.Message changed from "Your handset is not
provisioned for this network. " to "Falha de Registro na Rede JRTelecom"
```

Fonte: A autora.

Outra opção é desativar a mensagem de falha de registro na rede. Para isso, executa-se o comando:

```
OpenBTS> unconfig Control.LUR.FailedRegistration.Message
```

Como visto na Figura 3.5.3, a mensagem de registro na rede *Control.LUR.NormalRegistration.Message* encontra-se desativada por padrão. Essa mensagem é enviada ao aparelho celular sempre que ele é registrado com sucesso na rede. Para adicionar a mensagem “Bem-vindo a Rede JRTelecom” como uma mensagem de registro na rede executa-se o comando a seguir (Figura 3.5.5).

```
OpenBTS> config Control.LUR.NormalRegistration.Message Bem-
vindo a Rede JRTelecom
```

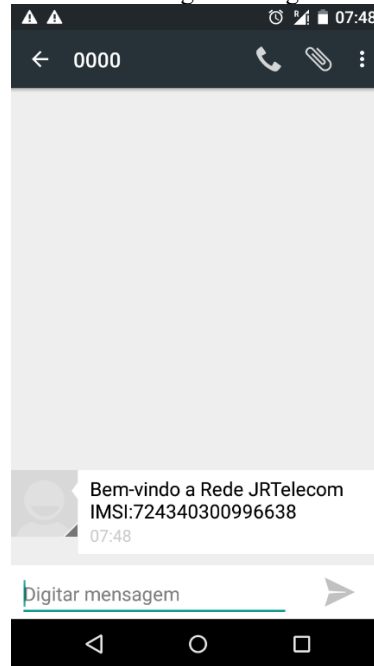
Figura 3.5.5 - Adicionando mensagem de registro na rede.

```
OpenBTS> config Control.LUR.NormalRegistration.Message Bem-vindo a Rede JRTelecom
Control.LUR.NormalRegistration.Message changed from "" to "Bem-vindo a Rede JRTelecom"
```

Fonte: A autora.

A Figura 3.5.6 mostra a mensagem de registro na rede sendo exibida em um aparelho celular logo após ele ser registrado com sucesso na rede JRTelecom.

Figura 3.5.6 - Mensagem de registro na rede.



Fonte: A autora.

3.6 Ajustes para Otimização do Desempenho da Rede

3.6.1 Causas de Rejeição de Assinantes

Para um melhor desempenho da rede, é possível adaptar o *OpenBTS* de acordo com o ambiente em que é a rede implantada. Para isso há diversos controles no *software OpenBTS* [2].

É preciso lidar com os aparelhos celulares não assinantes, uma vez que todos os aparelhos celulares que podem ver a rede tentarão ingressar nela, gerando LURs que serão rejeitadas. Esse tráfego de LURs sobrecarrega o sistema.

Ao executar o comando *tmsis*, a Figura 3.6.1.1 mostra um aparelho celular de IMEI 354463051143190 que não é um assinante da rede, pois a coluna AUTH está preenchida com o número zero. Esse aparelho aparece na lista justamente porque ele tentou ingressar na rede.

Figura 3.6.1.1 - Tentativa de ingresso na rede por um aparelho celular não assinante.

```
OpenBTS> tmsis
```

IMSI	TMSI	IMEI	AUTH	CREATED	ACCESSED	TMSI_ASSIGNED
724312921020978	0xbe1c1	352807062050680	1	34m	19m	1
724340300996638	0x1c456	354988059333580	1	41m	22m	1
724340301839223	0xae25	354463051143190	0	29m	29m	0

Fonte: A autora.

A chave *Control.LUR.UnprovisionedRejectCause* define qual causa usar quando um assinante desconhecido tenta ingressar na rede, ou seja, quando o IMSI não foi encontrado no banco de dados do registro de assinantes. Há diversas maneiras diferentes para rejeitar as solicitações de LUR dos aparelhos não assinantes.

Existe uma outra chave chamada *Control.LUR.404RejectCause* que define qual causa usar quando um assinante conhecido tenta entrar na rede, mas ocorre falha na autenticação.

Tanto para assinantes como para não assinantes da rede, as causas de rejeição instruem os aparelhos celulares a se afastarem por um longo período de tempo, mas não os fazem desistir de ingressar na rede.

Executa-se o comando a seguir para visualizar as duas causas de rejeições utilizadas nas duas chaves *Control.LUR.404RejectCause* e *Control.LUR.UnprovisionedRejectCause* (Figura 3.6.1.2).

```
OpenBTS> config RejectCause
```

Figura 3.6.1.2 - Causas de rejeições padrão do *OpenBTS*.

```
OpenBTS> config RejectCause
Control.LUR.404RejectCause 0x04 [default]
Control.LUR.UnprovisionedRejectCause 0x04 [default]
```

Fonte: A autora.

Para os aparelhos não assinantes executa-se o comando a seguir para verificar todas as causas de rejeições disponíveis (Figura 3.6.1.3).

```
OpenBTS> config Control.LUR.UnprovisionedRejectCause
```

Figura 3.6.1.3 - Causas de Rejeição disponíveis para não assinantes.

```

OpenBTS> config Control.LUR.UnprovisionedRejectCause
Control.LUR.UnprovisionedRejectCause 0x0C
- description:      Reject cause for location updating failures for unprovis
ioned phones, that is, the IMSI was not found in the Registrar database. The
SIP result code from the Registrar in this case is 401. Reject causes come
from GSM 04.08 10.5.3.6. Reject cause 0x02 or 0x04 is usually the right one.
- type:             multiple choice
- default value:    0x04
- visibility level: customer warn - a warning will be presented and confirma
tion required before changing this sensitive setting
- static:           0
- valid values:     0x02 = IMSI unknown in HLR
- valid values:     0x04 = IMSI unknown in VLR
- valid values:     0x05 = IMEI not accepted
- valid values:     0x0B = PLMN not allowed
- valid values:     0x0C = Location Area not allowed
- valid values:     0x0D = Roaming not allowed in this location area
- valid values:     0x11 = Network failure
- valid values:     0x16 = Congestion
- valid values:     0x20 = Service option not supported
- valid values:     0x21 = Requested service option not subscribed
- valid values:     0x22 = Service option temporarily out of order
- valid values:     0x26 = Call cannot be identified
- valid values:     0x30 = Retry upon entry into a new cell
- valid values:     0x5F = Semantically incorrect message
- valid values:     0x60 = Invalid mandatory information
- valid values:     0x61 = Message type non-existent or not implemented
- valid values:     0x62 = Message type not compatible with the protocol sta
te
- valid values:     0x63 = Information element non-existent or not implement
ed
- valid values:     0x64 = Conditional IE error
- valid values:     0x65 = Message not compatible with the protocol state
- valid values:     0x6F = Unspecified protocol error

```

Fonte: A autora.

Para os aparelhos assinantes executa-se o comando a seguir para verificar todas as causas de rejeições disponíveis (Figura 3.6.1.4).

```
OpenBTS> config Control.LUR.404RejectCause
```

Figura 3.6.1.4 - Causas de Rejeição disponíveis para assinantes.

```

OpenBTS> config Control.LUR.404RejectCause
Control.LUR.404RejectCause 0x0C
- description:      Reject cause for location updating failures for phones t
hat fail authentication. The SIP result code from the Registrar in this case
is 404. Reject causes come from GSM 04.08 10.5.3.6. Reject cause 0x02 or 0
x04 is usually the right one.
- type:             multiple choice
- default value:    0x04
- visibility level: customer warn - a warning will be presented and confirma
tion required before changing this sensitive setting
- static:           0
- valid values:     0x02 = IMSI unknown in HLR
- valid values:     0x04 = IMSI unknown in VLR
- valid values:     0x05 = IMEI not accepted
- valid values:     0x0B = PLMN not allowed
- valid values:     0x0C = Location Area not allowed
- valid values:     0x0D = Roaming not allowed in this location area
- valid values:     0x11 = Network failure
- valid values:     0x16 = Congestion
- valid values:     0x20 = Service option not supported
- valid values:     0x21 = Requested service option not subscribed
- valid values:     0x22 = Service option temporarily out of order
- valid values:     0x26 = Call cannot be identified
- valid values:     0x30 = Retry upon entry into a new cell
- valid values:     0x5F = Semantically incorrect message
- valid values:     0x60 = Invalid mandatory information
- valid values:     0x61 = Message type non-existent or not implemented
- valid values:     0x62 = Message type not compatible with the protocol sta
te
- valid values:     0x63 = Information element non-existent or not implement
ed
- valid values:     0x64 = Conditional IE error
- valid values:     0x65 = Message not compatible with the protocol state
- valid values:     0x6F = Unspecified protocol error

```

Fonte: A autora.

Como observado na Figura 3.6.1.2, por padrão são utilizadas causas de rejeição (0x04) para as chaves *Control.LUR.404RejectCause* e *Control.LUR.UnprovisionedRejectCause*. De acordo com as Figuras 3.6.1.3 e 3.6.1.4, a causa de rejeição (0x04 = *IMSI unknown in VLR*) é utilizada para IMSI desconhecido no VLR (Registro de Localização do Visitante).

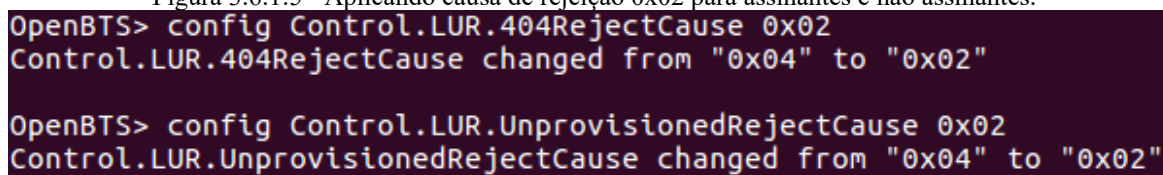
A causa de rejeição (0x04) faz com que o aparelho celular continue tentando registrar na rede várias vezes (em um temporizador de 15 segundos) e depois irá parar essas tentativas de registro na rede [23].

A causa de rejeição (0x02 = *IMSI unknown in HLR*) pode ser utilizada para rejeitar IMSI desconhecido no HLR (Registro de Local de Origem). Utilizando essa causa de rejeição, o aparelho celular mostrará a mensagem de falha de autenticação na rede e não tentará se registrar novamente nesta rede ou em qualquer outra rede até que seja desligado [24].

Assim, com o objetivo de fazer com que o tráfego LUR não sobrecarregue a rede, executa-se os comandos a seguir para utilizar uma causa de rejeição (0x02) para as chaves *Control.LUR.404RejectCause* e *Control.LUR.UnprovisionedRejectCause*. A Figura 3.6.1.5 mostra a execução desses comandos.

```
OpenBTS> config Control.LUR.404RejectCause 0x02
OpenBTS> config Control.LUR.UnprovisionedRejectCause 0x02
```

Figura 3.6.1.5 - Aplicando causa de rejeição 0x02 para assinantes e não assinantes.



```
OpenBTS> config Control.LUR.404RejectCause 0x02
Control.LUR.404RejectCause changed from "0x04" to "0x02"

OpenBTS> config Control.LUR.UnprovisionedRejectCause 0x02
Control.LUR.UnprovisionedRejectCause changed from "0x04" to "0x02"
```

Fonte: A autora.

3.6.2 Área de Cobertura

Uma outra maneira de fazer com que os aparelhos ignorem a rede é impedi-los de vê-la, isso é feito encolhendo a área de cobertura utilizável.

Em geral, a área de cobertura para serviços de SMS e registro é aproximadamente quatro vezes maior do que a dos serviços de voz, pois os quadros perdidos podem ser retransmitidos sem interrupção dos serviços [2].

Conforme descrito na seção 3.2 (Configurações Básicas do *OpenBTS*), através do comando *power*, o *OpenBTS* ajusta a atenuação da potência de transmissão de *downlink* para expandir ou contrair a área de cobertura. A atenuação da potência de transmissão de *downlink*, por padrão, é configurada em -10 dB, ou seja, o *downlink* é transmitido com menos 10 dB de potência. Aumentar essa atenuação significa diminuir a potência de transmissão de *downlink* e, consequentemente, diminuir a área de cobertura. Portanto, isso não deve ser feito.

Para maximizar a área de cobertura, o *downlink* deve ser transmitido com máxima potência, ou seja, não deve haver atenuação. Assim o comando *power* deve ser configurado como zero, para isso, execute o comando abaixo. A Figura 3.6.2.1 mostra a execução desse comando.

```
OpenBTS> power 0
```

Figura 3.6.2.1 - Comando para que o *downlink* seja transmitido com a máxima potência

```
OpenBTS> power 0
current downlink power 0 dB wrt full scale
```

Fonte: A autora.

Há um outro controle de potência disponível. Como os aparelhos celulares transmitem para a estação rádio base utilizando diferentes níveis de potência, ao alcançar a estação rádio base a potência recebida de todos esses aparelhos é aproximadamente igual em intensidade.

O aparelho celular não tem controle sobre qual nível de potência utilizar. A estação rádio base é quem controla os níveis de potência utilizados e disponíveis para os aparelhos celulares. As chaves *MS.Power* permitem limitar os valores de potência que esses aparelhos podem utilizar. Executa-se o comando a seguir para verificar os valores dessas chaves. Foram mantidos os valores padrões vistos na Figura 3.6.2.1.

```
OpenBTS> config MS.Power
```

Figura 3.6.2.2 - Valores padrão das chaves *MS.Power*.

```
OpenBTS> config MS.Power
GSM.MS.Power.Damping 75 [default]
GSM.MS.Power.Max 33 [default]
GSM.MS.Power.Min 5 [default]
```

Fonte: A autora.

Conforme visto na Figura 3.6.2.1, o parâmetro *GSM.MS.Power.Damping*, por padrão, é configurado em 75 %. Esse parâmetro define em porcentagem o valor de amortecimento para o *loop* de controle de potência do aparelho celular. A faixa de valores para esse parâmetro é de 0 a 100 %. O valor 100 faz ignorar completamente a RSSI e a SNR. Já um valor igual a zero, faz com que a potência do aparelho celular mude instantaneamente com base na RSSI ou na SNR, o que não é aconselhável, pois provoca oscilações de potência.

O parâmetro *GSM.MS.Power.Max*, por padrão, é configurado em 33 dBm. Esse parâmetro representa o nível máximo de potência para os aparelhos celulares, podendo ser atribuído valores de 0 a 39 dBm. Já o parâmetro *GSM.MS.Power.Min*, por padrão, é configurado em 5 dBm. Esse parâmetro representa o nível mínimo de potência para os aparelhos celulares, podendo ser atribuído valores de 0 a 39 dBm.

Escolher fisicamente a área de cobertura pode não ser suficiente para estabilizar a rede sob uma carga de uso pesado. Assim, há um outro parâmetro bastante útil, o parâmetro GSM.MS.TA.MAX.

TA significa *Timing Advance*, esse é um método utilizado pelo GSM para compensar aparelhos celulares que estão muito distantes da estação rádio base. Quanto mais longe estiver o aparelho celular, maior deverá ser o valor de TA.

O valor TA, medido em período de símbolo, informa ao aparelho celular que é preciso transmitir suas rajadas de rádio mais cedo para que cheguem à estação rádio base diretamente dentro do *time slot* alocado. Executou-se o comando a seguir para verificar os valores da chave GSM.MS.TA (Figura 3.6.2.3).

```
OpenBTS> config GSM.MS.TA
```

Figura 3.6.2.3 - Valores padrões para a chave GSM.MS.TA.

```
OpenBTS> config GSM.MS.TA
GSM.MS.TA.Damping 50      [default]
GSM.MS.TA.Max 62         [default]
```

Fonte: A autora.

Os valores válidos para o parâmetro GSM.MS.TA.MAX estão entre 1 e 62. O valor unitário de TA corresponde a uma distância de aproximadamente 550 metros [2]. Executou-se o comando a seguir para configurar o parâmetro GSM.MS.TA.MAX para o valor 10 (Figura 3.6.2.4). Isso faz com que o *OpenBTS* ignore qualquer rajada de rádio com um TA maior que 10, ou seja, de aparelhos localizados a mais de 5,5 quilômetros de distância da estação rádio base. A Figura 3.6.2.5 mostra o novo valor configurado para o parâmetro GSM.MS.TA.MAX.

```
OpenBTS> config GSM.MS.TA.Max 10
```

Figura 3.6.2.4 - Configurando parâmetro GSM.MS.TA.Max para 10.

```
OpenBTS> config GSM.MS.TA.Max 10
GSM.MS.TA.Max changed from "62" to "10"
```

Fonte: A autora.

Figura 3.6.2.5 - Novo valor configurado para o parâmetro GSM.MS.TA.MAX.

```
OpenBTS> config GSM.MS.TA
GSM.MS.TA.Damping 50      [default]
GSM.MS.TA.Max 10
```

Fonte: A autora.

3.6.3 Distorção do Sinal

A distorção do sinal depende muito do terreno em torno da implantação da rede móvel. O *OpenBTS* é configurado para ser bastante completo em atenuar as distorções *multipath*. O *OpenBTS* faz isso analisando um grande número de períodos de símbolos ao tentar cancelar as distorções. Porém, isso torna o trabalho computacional bastante pesado [2].

Para implantações da rede em terrenos abertos, sem edifícios ou árvores, ou até mesmo em implantações com áreas de cobertura menores, pode-se reduzir a carga computacional ajustando a chave *GSM.Radio.MaxExpectedDelaySpread*. Essa chave determina quantos períodos de símbolo são examinados, os valores válidos são de 1 a 4. O *delay*, esperado no pior caso, corresponde a 3,7 microssegundos ou 1,1 quilômetro por unidade de período de símbolo [2]. Executa-se o comando a seguir para verificar o valor padrão dessa chave (Figura 3.6.3.1).

```
OpenBTS> config Delay
```

Figura 3.6.3.1 - Valor padrão da chave *GSM.Radio.MaxExpectedDelaySpread*.

```
OpenBTS> config Delay
GSM.Radio.MaxExpectedDelaySpread 4      [default]
```

Fonte: A autora.

Como a rede está implantada em um ambiente de laboratório, executa-se os comandos abaixo para alterar o valor da chave *GSM.Radio.MaxExpectedDelaySpread* de 4 para 1 e depois mostrar o novo valor dessa chave. A execução desses comandos é mostrada na Figura 3.6.3.2.

```
OpenBTS> config GSM.Radio.MaxExpectedDelaySpread 1
OpenBTS> config Delay
```

Figura 3.6.3.2 - Alterando o valor da chave *GSM.Radio.MaxExpectedDelaySpread*.

```
OpenBTS> config GSM.Radio.MaxExpectedDelaySpread 1
GSM.Radio.MaxExpectedDelaySpread changed from "4" to "1"

OpenBTS> config Delay
GSM.Radio.MaxExpectedDelaySpread 1
```

Fonte: A autora.

3.7 Serviço *OpenRegistration*

O *OpenRegistration* é um serviço disponível no *OpenBTS* que permite aos não assinantes ingressarem na rede. Como não é necessário um administrador para criar contas de assinantes, esse serviço torna-se mais fácil de ser implantado, e além disso é muito útil para os usuários [2].

O *OpenRegistration* é apropriado para casos onde a rede é necessária temporariamente, como por exemplo em grandes festivais, lugares remotos, casos de emergências [2].

Para verificar as configurações do serviço *OpenRegistration* executa-se o seguinte comando:

```
OpenBTS> config OpenRegistration
```

Figura 3.7.1 - Configurações padrões do *OpenBTS* para o serviço *OpenRegistration*.

```
OpenBTS> config OpenRegistration
Control.LUR.OpenRegistration (disabled)      [default]
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]
```

Fonte: A autora.

Para ativar o serviço *OpenRegistration* a chave *Control.LUR.OpenRegistration* mostrada na Figura 3.7.1 deve estar definida por uma expressão regular. Essa expressão regular é que define quais IMSIs receberão acesso à rede [2]. O Quadro 3.7.1 apresenta algumas expressões regulares e seus respectivos efeitos.

Quadro 3.7.1 - Expressões regulares e seus efeitos

Expressão Regular	Efeitos no <i>OpenRegistration</i>
.	Aceitar qualquer IMSI
^724	Aceitar qualquer IMSI que começa com “724”, o MCC do Brasil [24]
724340300996638	Aceitar apenas o IMSI 724340300996638
0	Aceitar qualquer IMSI que contém um “0”
1	Aceitar qualquer IMSI que contém um “1”
1234	Aceitar qualquer IMSI que termina em 1234

Fonte: IEDEMA, 2015, p. 68 (Adaptado)

Executa-se o comando a seguir para ativar o serviço *OpenRegistration* e aceitar qualquer IMSI que desejar ingressar na rede através desse serviço. A Figura 3.7.2 mostra a execução desse comando.

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

Figura 3.7.2 - Ativando o serviço *OpenRegistration*.

```
OpenBTS> config Control.LUR.OpenRegistration .*
Control.LUR.OpenRegistration changed from "" to ".*"
```

Fonte: A autora.

Como mostrado na Figura 3.7.3, executou-se o comando a seguir para alterar a mensagem de boas-vindas do serviço *OpenRegistration* para “Bem-vindo ao OpenRegistration da JRTelecom. Disque 101.”

```
OpenBTS> config Control.LUR.OpenRegistration.Message Bem-vindo
ao OpenRegistration da JRTelecom. Disque 101.
```

Figura 3.7.3 - Alterando a mensagem de boas-vindas do serviço *OpenRegistration*.

```
OpenBTS> config Control.LUR.OpenRegistration.Message Bem-vindo ao OpenRegistrati
on da JRTelecom. Disque 101.
Control.LUR.OpenRegistration.Message changed from "" to "Bem-vindo ao OpenRegi
stration da JRTelecom. Disque 101."
```

Fonte: A autora.

Executa-se o comando “*config OpenRegistration*” para visualizar as novas configurações aplicadas para o serviço *OpenRegistration*. O resultado é mostrado na Figura 3.7.4.

Figura 3.7.4 - Novas configurações para o serviço *OpenRegistration*.

```
OpenBTS> config OpenRegistration
Control.LUR.OpenRegistration .*
Control.LUR.OpenRegistration.Message Bem-vindo ao OpenRegistration da JRTelecom. Disque 101.
Control.LUR.OpenRegistration.Reject (disabled) [default]
Control.LUR.OpenRegistration.ShortCode 101 [default]
```

Fonte: A autora.

A princípio, um usuário do *OpenRegistration* pode fazer ligações, entretanto, não é possível receber ligações de nenhum assinante da rede, uma vez que não há nenhum MSISDN atribuído àquele usuário [2].

O componente *smqueue* tem um manipulador de *shortcode* para o número 101 que, por padrão, é definido na chave *Control.LUR.OpenRegistration.ShortCode* mostrada na Figura 3.7.1 [2].

Os usuários do *OpenRegistration* são instruídos a procurarem ajuda ligando para o número 101, então é necessário ter alguém na rede atribuído ao MSISDN 101 [2]. Para isso, cadastrou-se o usuário “Assistência”, MSISDN 101 e IMSI 724109393365788 assim como mostra o Quadro 3.4.1 disponível na seção 3.4.

Caso o usuário do *OpenRegistration* deseje ter um MSISDN atribuído a ele, é possível provisioná-lo interagindo com o número 101 via SMS, isto é chamado de auto provisionamento.

São os parâmetros *SC.Register* no *smqueue* que controlam qual *shortcode* é utilizado, como os SMSs devem ser escritos e como deve ser o MSISDN para o usuário do *OpenRegistration*. Embora o *smqueue* não tenha uma interface de linha de comandos, o *nmcli.py* (descrito na seção 3.4) pode ser utilizado para ler e modificar esses parâmetros [2]. Para o presente trabalho, esses parâmetros não foram alterados. Entretanto, para acessar as configurações desses parâmetros e modificá-las utiliza-se os seguintes comandos:

```
$ ./nmcli.py smqueue config read
$ ./nmcli.py smqueue config update Configuration.Key.Name
valorDesejado
```

De acordo com [25], logo após um usuário ingressar na rede pelo serviço do *OpenRegistration*, esse usuário receberá uma mensagem de boas-vindas do MSISDN 101. Para realizar o auto provisionamento, o usuário deve enviar um MSISDN desejado de 10 dígitos para o número 101. Posteriormente, dentro de alguns minutos, o aparelho celular receberá um SMS indicando se o auto provisionamento foi realizado com sucesso.

Para desativar o serviço *OpenRegistration*, revertendo o *OpenBTS* de modo a utilizar o bando de dados do registro de assinantes para conceder ou negar acessos ao invés de padrões de IMSIs, executa-se os comandos a seguir:

```
OpenBTS> unconfig Control.LUR.OpenRegistration
OpenBTS> unconfig Control.LUR.OpenRegistration.Reject
```

3.8 Expansão da Rede

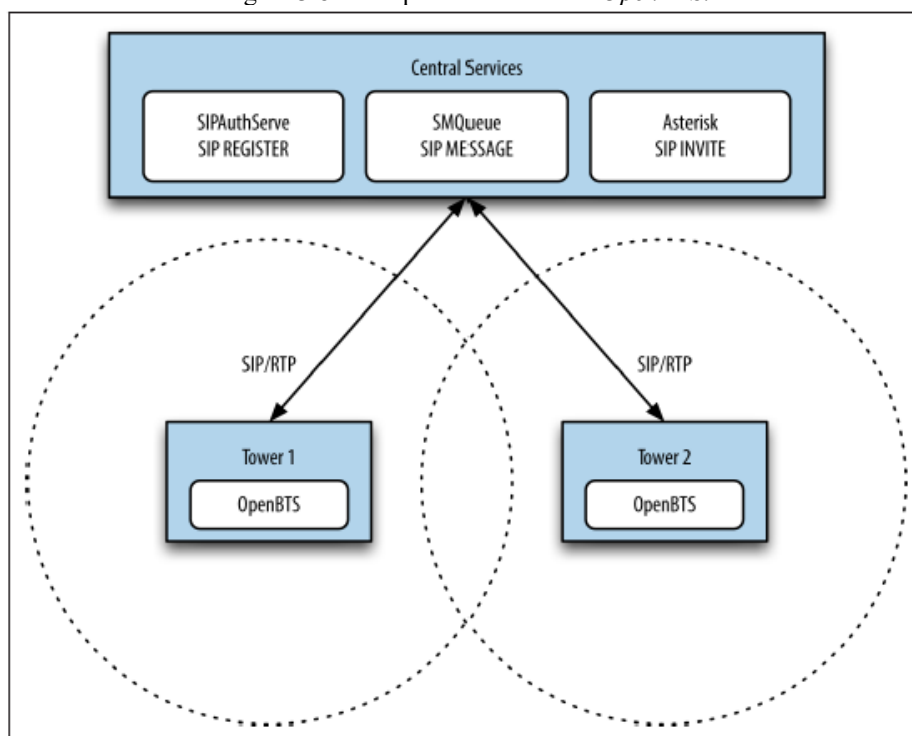
O *OpenBTS* utiliza a mesma pilha de software para abranger tanto pequenas áreas de cobertura como grandes [2].

O trabalho aqui realizado implementa uma única estação rádio base. No entanto, para fazer cobertura de uma grande área, a rede móvel precisa de várias estações rádio base. Assim essa rede móvel suportará a mobilidade e o *handover* como qualquer rede comercial [2].

O *OpenBTS* permite a implementação de uma rede móvel com diversas estações rádio base. Por uma questão de simplicidade, as estações rádio base passarão a ser chamadas de “torre” e os componentes *sipauthserve*, *smqueue* e *Asterisk* serão chamados de Serviços Centrais [2].

Para expansão da rede, deverá haver várias instalações de “torres” que executarão o *OpenBTS*, e uma única instalação de Serviços Centrais. A Figura 3.8.1 mostra a topologia dessa rede.

Figura 3.8.1 - Expansão da rede no *OpenBTS*.



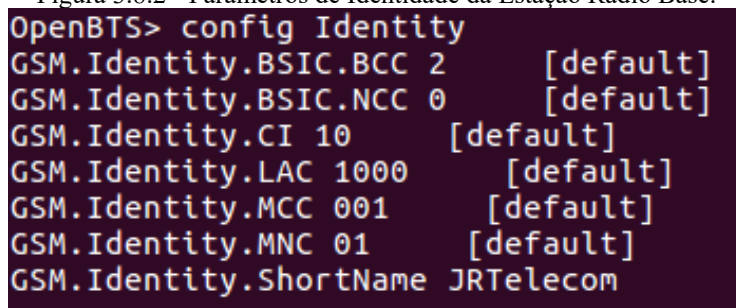
Fonte: IEDEMA, 2015, p. 51.

Para permitir que as várias áreas de cobertura compartilhem o mesmo banco de dados e as configurações do registro de assinantes, as “torres” comunicarão por IP com os Serviços Centrais. Assim, o *OpenBTS* deixará de se comunicar localmente com o *sipauthserve*, *Asterisk* e *smqueue*, e passará a se comunicar com esses componentes por meio de uma rede IP.

O comando a seguir é executado para visualizar os parâmetros de identidade da única “torre”, até então utilizada. A Figura 3.8.2 mostra a execução desse comando.

OpenBTS> config Identity

Figura 3.8.2 - Parâmetros de Identidade da Estação Rádio Base.



```

OpenBTS> config Identity
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 001         [default]
GSM.Identity.MNC 01          [default]
GSM.Identity.ShortName JRTelecom
  
```

Fonte: A autora.

Alguns das chaves mostradas na Figura 3.8.2 devem ser únicas para cada “torre” e outras devem ser idênticas para as “torres” vizinhas.

As chaves que devem ser idênticas para cada torre são:

- 1) *GSM.Identity.MCC*;
- 2) *GSM.Identity.MNC*;
- 3) *GSM.Identity.BSIC.NCC*;
- 4) *GSM.CellSelection.NCCsPermitted*;
- 5) *GSM.Identity.ShortName*.

As chaves *GSM.Identity.MCC* e *GSM.Identity.MNC* são a identificação de nível mais alto para qualquer rede. Essa identificação é atribuída por órgão regulamentadores [2].

A chave *GSM.Identity.MCC* indica em qual país está a rede, no *OpenBTS* é utilizado o valor 001. Já a chave *GSM.Identity.MNC* indica qual operadora presta o serviço, no *OpenBTS* é utilizado o valor 01 [2].

A chave *GSM.Identity.BSIC.NCC* é o Código de Cores da Rede (NCC), uma informação que os aparelhos celulares utilizam para determinar se eles podem obter acesso à rede. Todas as operadoras em uma determinada área devem ter códigos de cores exclusivos [2].

A chave *GSM.CellSelection.NCCsPermitted* é um sinalizador bit a bit, e não uma valor inteiro. Ela é utilizada para permitir NCCs de outras operadoras [2].

Embora a chave *GSM.Identity.ShortName* possa ser idêntica em todas as “torres”, não é aconselhável, pois com um nome exclusivo para cada torre é fácil saber em qual “torre” o aparelho celular se encontra enquanto é movimentado [2].

As chaves que devem ser únicas para cada torre são:

- 1) *GSM.Identity.LAC*;

- 2) *GSM.Identity.CI*;
- 3) *GSM.Identity.BSIC.BCC*;
- 4) *GSM.Radio.CO*.

A chave *GSM.Radio.CO* é o ARFCN, ou seja, o par de radiofrequências utilizado na “torre”. Se as torres fisicamente adjacentes estiverem utilizando a mesma frequência, haverá interferência e, portanto, o serviço a qualquer usuário na região de sobreposição do sinal será negado [2].

Além de terem que ser únicos, os ARFCNs também não podem ser numericamente adjacentes. Recomenda-se um intervalo de pelo menos um ARFCN entre o ARFCN de uma “torre” e o ARFCN de outra “torre”. Por exemplo, a “torre” 1 utiliza o ARFCN 100 e a “torre” 2 utiliza o ARFCN 102 [2].

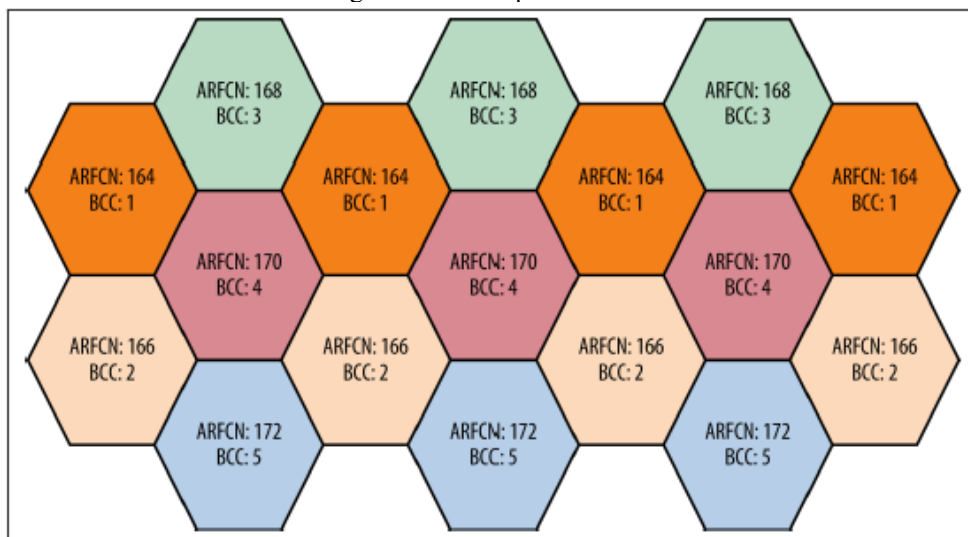
A chave *GSM.Identity.BSIC.BCC* sinaliza o Código de Cores da Estação Base (BCC), um campo de 3 bits que permite sete valores exclusivos [2].

Como os ARFCNs licenciados e o BCC são muito limitados numericamente, um código de cores é utilizado para atribuir valores à “torre” caso seja necessário muitas “torres” [2].

Um mapa de cores mostra a localização geográfica de todas as “torres” e atribui a cada uma delas uma cor. Cada cor corresponde a um número ARFCN e um par BCC [2].

Se não tiver duas “torres” adjacentes com a mesma cor, os serviços de mobilidade e *handover* não serão prejudicados pelo confronto de ARFCNs ou de BCCs. A Figura 3.8.3 mostra um exemplo de mapa de cores [2].

Figura 3.8.3 - Mapa de cores.



Fonte: Fonte: IEDEMA, 2015, p. 56.

Em uma rede GSM tradicional, todas as estações rádio base de uma determinada região geográfica têm o mesmo LAC (Código de Área Local). Entretanto, no *OpenBTS* há um LAC exclusivo para cada estação rádio base, assim as operações LURs podem ser executadas ao mover entre estações rádio base. Então, a chave *GSM.Identity.LAC* deve ser única para cada “torre”.

Quando um aparelho se move através dos limites da torre, percebe que o Código de Área de Localização (LAC) está mudando e deve realizar outra LUR. A LUR é convertida em um *SIP REGISTER*, e o *sipauthserve* pode atualizar o endereço IP da torre onde o aparelho celular está acessível [2].

Finalmente, a chave *GSM.Identity.CI* é a *Cell ID* (CI), e deve ser única em todas as torres.

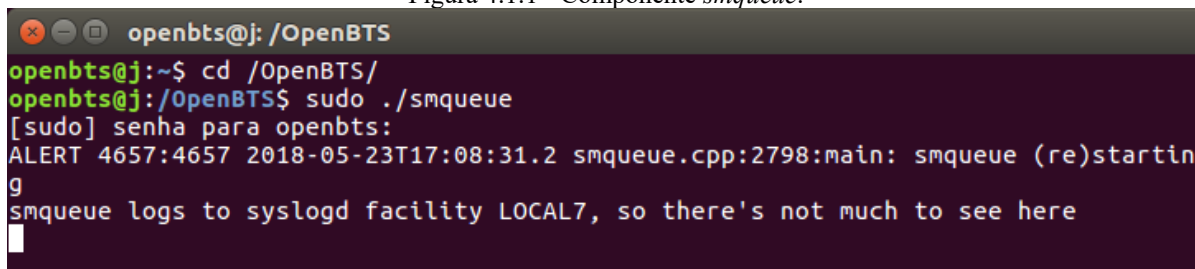
4 RESULTADOS E DISCUSSÃO

4.1 Teste de SMS

Para testar o serviço de SMS, deve-se iniciar em um outro terminal do *Linux* o componente *smqueue* (Figura 4.1.1). Este é responsável por receber, rotear e programar a entrega de mensagens SMS [2]. Executa-se os comandos a seguir para iniciá-lo:

```
$ cd /OpenBTS  
$ sudo ./smqueue
```

Figura 4.1.1 - Componente *smqueue*.

A terminal window titled 'openbts@j: /OpenBTS' with a dark purple background. The prompt is 'openbts@j:~\$'. The user enters 'cd /OpenBTS/'. The prompt changes to 'openbts@j:/OpenBTS\$'. The user enters 'sudo ./smqueue'. The prompt changes to '[sudo] senha para openbts:'. The user enters a password (indicated by dots). The terminal displays the output: 'ALERT 4657:4657 2018-05-23T17:08:31.2 smqueue.cpp:2798:main: smqueue (re)starting' and 'smqueue logs to syslogd facility LOCAL7, so there's not much to see here'.

```
openbts@j:~$ cd /OpenBTS/  
openbts@j:/OpenBTS$ sudo ./smqueue  
[sudo] senha para openbts:  
ALERT 4657:4657 2018-05-23T17:08:31.2 smqueue.cpp:2798:main: smqueue (re)starting  
smqueue logs to syslogd facility LOCAL7, so there's not much to see here
```

Fonte: A autora.

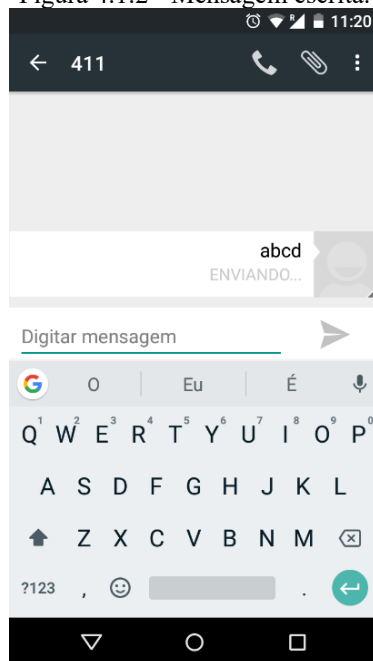
Inicialmente, para verificar o funcionamento correto do serviço de SMS disponível na rede JRTelecom, realiza-se um teste enviando um SMS para o número 411.

O número 411 é um manipulador de *shortcode* do *smqueue*. Sua função é retornar a mensagem que recebe juntamente com informações adicionais sobre a rede e a conta do assinante que foi utilizada [2].

Então, diretamente de um aparelho celular registrado na rede, no aplicativo padrão de SMS, deve-se escrever uma mensagem de texto e enviá-la para o número 411. É recomendado que o corpo da mensagem seja constituído por letras ou números sequenciais para ajudar a identificar se a mensagem retornada pelo número 411 está correta. Esse retorno deve ocorrer dentro de poucos segundos após o envio da mensagem.

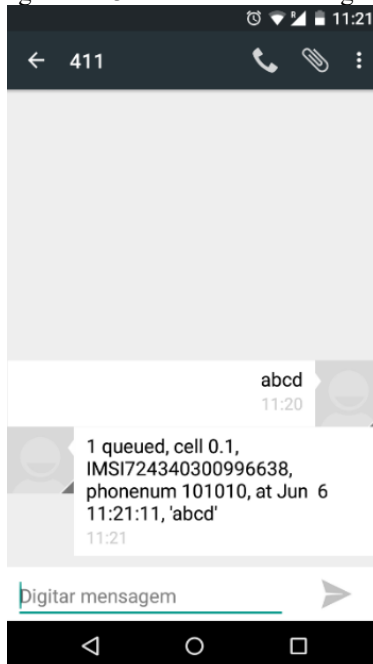
Para realizar esse teste, enviou-se a mensagem “abcd” para o número 411, originada no aparelho celular com chip de IMSI igual 724340300996638 (Assinante “Juliana”, MSISDN 101010). As Figuras 4.1.2 e 4.1.3 mostram respectivamente a mensagem escrita e o retorno dessa mensagem no aparelho celular.

Figura 4.1.2 - Mensagem escrita.



Fonte: A autora.

Figura 4.1.3 - Retorno da mensagem.



Fonte: A autora.

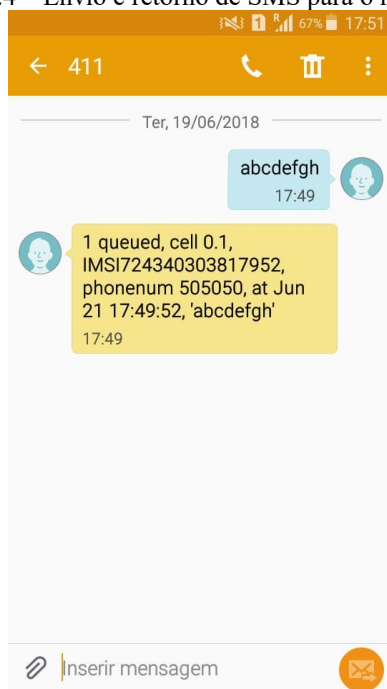
Cada parte da mensagem de retorno vista na Figura 4.1.3 tem um significado:

- 1) *1 queued*: significa que há uma mensagem enfileirada para entrega;
- 2) *cell 0.1*: significa que a estação base tem um fator de carga de 0,1;

- 3) IMSI724340300996638: significa que a mensagem foi recebida do IMSI 724340300996638
- 4) *phonenum* 101010: significa que a mensagem foi enviada pelo número de telefone MSISDN 101010;
- 5) *at Jun 6 11:21:11*: mensagem foi enviada em 6 de junho às 11:21:11 horas;
- 6) abcd: O corpo da mensagem é abcd.

Para repetir o teste anterior, enviou-se a mensagem “abcdefgh” para o número 411, originada no aparelho celular com chip de IMSI igual 724340303817952 (Assinante “Samsung1”, MSISDN 505050). O resultado é mostrado na Figura 4.1.4.

Figura 4.1.4 – Envio e retorno de SMS para o número 411.



Fonte: A autora.

As mensagens SMS também podem ser testadas diretamente do *OpenBTS* utilizando o comando *sendsms*. Este comando tem o formato:

```
OpenBTS> sendsms IMSI src# mensagem
```

No comando anterior, especifique o IMSI de destino, o número de origem da mensagem (*src#*) e o conteúdo da mensagem.

A Figura 4.1.5 mostra esse comando sendo executado para enviar uma mensagem direto do *OpenBTS*, para o IMSI 724340300996638, da assinante “Juliana”, originada do

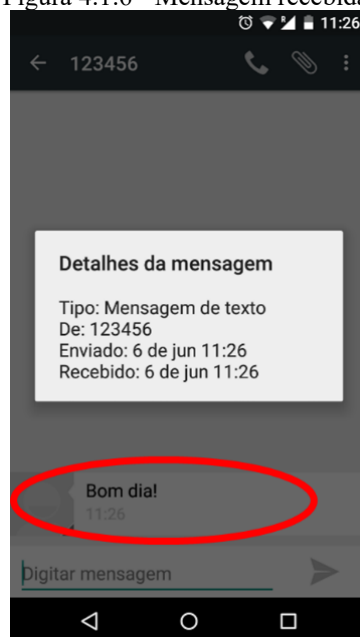
número fictício 123456. Esse número fictício não é número MSISDN de nenhum aparelho registrado na rede. A Figura 4.1.6 mostra o recebimento da mensagem do número fictício no aparelho celular.

Figura 4.1.5 - Utilizando comando *sendsms*.

```
OpenBTS> sendsms 724340300996638 123456 Bom dia!  
message submitted for delivery
```

Fonte: A autora.

Figura 4.1.6 - Mensagem recebida.



Fonte: A autora.

Para repetir o teste utilizando o comando *sendsms*, enviou-se a mensagem “Boa tarde!” originada do número fictício 123456 para o IMSI 724340303817952 (Assinante “Samsung1”, MSISDN 505050). As Figuras 4.1.7 e 4.1.8 mostram, respectivamente a mensagem enviada e o recebimento da mensagem no aparelho celular.

Figura 4.1.7 - Mensagem enviada.

```
OpenBTS> sendsms 724340303817952 123456 Boa tarde!  
message submitted for delivery
```

Fonte: A autora.

Figura 4.1.8 - Mensagem recebida.



Fonte: A autora.

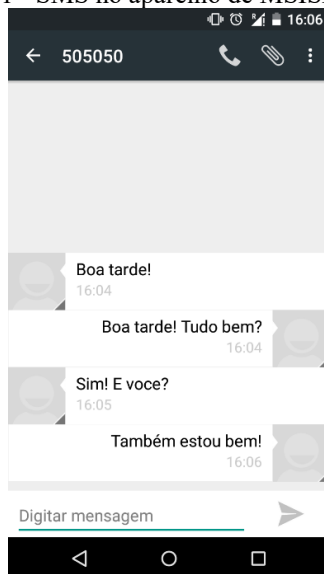
Mensagens SMS criadas utilizando o comando *sendsms* são enviadas diretamente pela interface aérea GSM para o aparelho. Elas não são roteadas pelo *smqueue*, portanto não podem ser remarcadas para envio posterior. Assim, se o telefone estiver inacessível, essas mensagens são simplesmente descartadas. Daí a importância do *smqueue*, pois ele reprograma a entrega das mensagens em caso de falha, o que é comum em comunicação *wireless* [2].

4.2 Teste de SMS entre dois aparelhos

Para testes de SMS entre dois aparelhos, é importante verificar se os números de origem estão corretos ao receber o SMS e se as respostas a esses SMS são encaminhadas de volta ao remetente original.

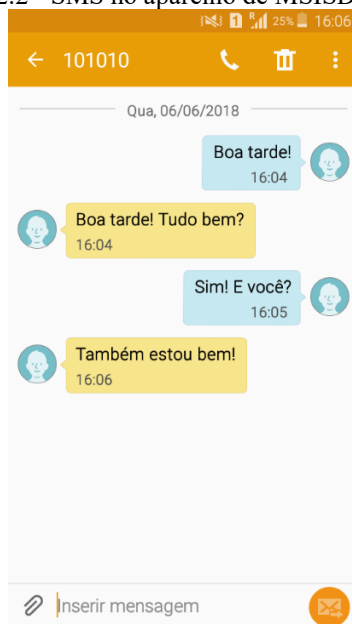
Foram trocados SMS entre os aparelhos celulares de MSISDN 101010 e 505050. A Figura 4.2.1 mostra as mensagens no aparelho de MSISDN 101010 e a Figura 4.2.2 as mensagens no aparelho de MSISDN 505050. Conforme observado nas Figuras 4.2.1 e 4.2.2, os números de origem das mensagens estão corretos e as respostas foram encaminhadas ao remetente original.

Figura 4.2.1 - SMS no aparelho de MSISDN 101010.



Fonte: A autora.

Figura 4.2.2 - SMS no aparelho de MSISDN 505050.



Fonte: A autora.

Para repetir o teste de SMS entre dois aparelhos, foram trocados SMS entre os aparelhos de MSISDN 303030 e MSISDN 505050. A Figura 4.2.3 mostra as mensagens no aparelho de MSISDN 303030 e a Figura 4.2.4 as mensagens no aparelho de MSISDN 505050. Conforme observado nas Figuras 4.2.3 e 4.2.4, os números de origem das mensagens estão corretos e as respostas foram encaminhadas ao remete original.

Figura 4.2.3 - SMS no aparelho de MSISDN 303030.



Fonte: A autora.

Figura 4.2.4 - SMS no aparelho de MSISDN 505050.



Fonte: A autora.

É possível monitorar vários tipos de eventos que ocorrem no *OpenBTS*, para isso é utilizado o comando *stats*. Cada tipo de evento é um nome de uma chave em um banco de dados

SQLite3 e o valor de cada chave corresponde ao número de vezes que esse evento aconteceu desde a última vez que as estatísticas do banco de dados foi limpa [2].

Para observar os eventos que ocorrem no *OpenBTS* quando um SMS é enviado, executa-se o comando a seguir para limpar o banco de dados e então obter uma contagem inicial conhecida dos eventos de interesse.

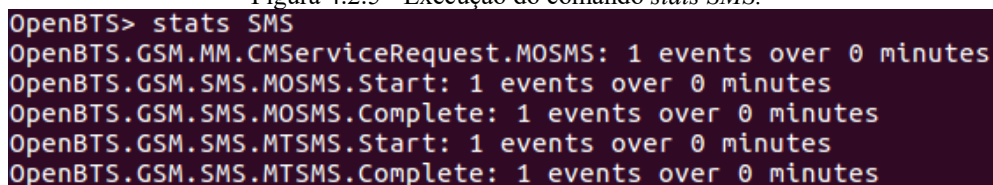
```
OpenBTS> stats clear
```

Após o comando anterior, enviou-se um SMS de um aparelho celular para outro. Posteriormente, o comando a seguir foi executado para pesquisar por eventos relacionados a este

SMS enviado. O resultado é mostrado na Figura 4.2.5.

```
OpenBTS> stats SMS
```

Figura 4.2.5 - Execução do comando *stats SMS*.



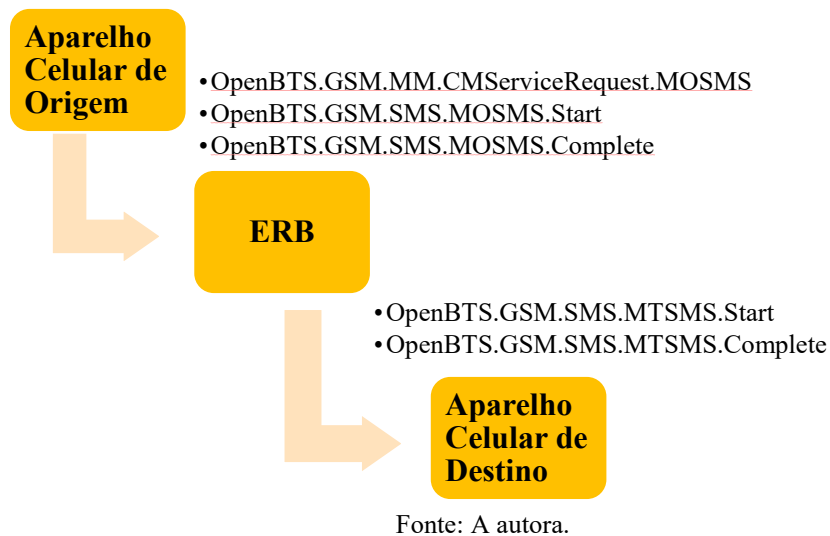
```
OpenBTS> stats SMS
OpenBTS.GSM.MM.CMServiceRequest.MOSMS: 1 events over 0 minutes
OpenBTS.GSM.SMS.MOSMS.Start: 1 events over 0 minutes
OpenBTS.GSM.SMS.MOSMS.Complete: 1 events over 0 minutes
OpenBTS.GSM.SMS.MTSMS.Start: 1 events over 0 minutes
OpenBTS.GSM.SMS.MTSMS.Complete: 1 events over 0 minutes
```

Fonte: A autora.

De acordo com as informações disponíveis no próprio *software OpenBTS*, a chave *OpenBTS.GSM.MM.CMServiceRequest.MOSMS* mostra que um aparelho celular sinalizou para a estação rádio base que desejava enviar um SMS originado no MOSMS (*Mobile Originated SMS*). As chaves *OpenBTS.GSM.SMS.MOSMS.Start* e *OpenBTS.GSM.SMS.MOSMS.Complete* mostram que um MOSMS foi iniciado e concluído. Essas três primeiras chaves estão relacionadas à transmissão do SMS originado no aparelho celular com destino à estação rádio base [2].

As duas últimas chaves *OpenBTS.GSM.SMS.MTSMS.Start* e *OpenBTS.GSM.SMS.MTSMS.Complete* mostram que um MTSMS (*Mobile Terminated SMS*) foi iniciado e concluído. Essas duas últimas chaves estão relacionadas à transmissão do SMS originado na estação rádio base com destino ao outro aparelho celular [2]. Para compreender melhor o processo de envio de um SMS entre dois aparelhos celulares, construiu-se o fluxograma mostrado na Figura 4.2.6.

Figura 4.2.6 - Processo de envio de um SMS entre dois aparelhos celulares.



4.3 Teste de Chamadas

Para testar o serviço de Voz, deve-se iniciar em um outro terminal do *Linux* o componente *Asterisk* (Figura 4.3.1). Execute o comando a seguir para iniciá-lo:

```
$ sudo asterisk -r
```

Figura 4.3.1 - Componente *Asterisk*.

```

openbts@j: ~
openbts@j:~$ sudo asterisk -r
Privilege escalation protection disabled!
See https://wiki.asterisk.org/wiki/x/1gKfAQ for more details.
Asterisk 11.7.0.5, 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 11.7.0.5 currently running on j (pid = 19251)
j*CLI>
  
```

Fonte: A autora.

O pacote *rangeasterisk-configs* define algumas extensões de teste para verificar alguns aspectos do serviço de voz. Uma extensão é um número de telefone interno, inacessível ao mundo exterior [2].

Uma das extensões de teste disponíveis no *Asterisk* é a *Test Tone Call* (2602). Diretamente do aparelho celular registrado na rede, no teclado padrão de chamadas de voz, deve-se ligar para o número 2602. Ao fazer isso, ouve-se um tom e percebe-se que esse tom sofre alterações [2].

As alterações no tom são devido à falta de informações no caminho do fluxo de voz do *downlink*, semelhante à perda de pacotes. Essa extensão é utilizada para testar a qualidade de *downlink*. Em redes de produção, uma perda de *downlink* de 3% é normal e, perdas de 5% a 7% ainda são capazes de fornecer uma conversa compreensível [2].

Uma ligação bem-sucedida para o número 2602 também confirma os seguintes aspectos sobre a rede:

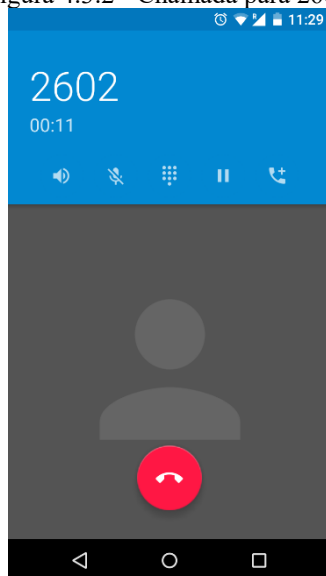
- 1) O *Asterisk* foi executado com sucesso;
- 2) O roteamento de chamadas funcionou conforme o esperado;
- 3) O áudio de *downlink* é funcional.

Outra extensão de teste disponível no *Asterisk* é a *Echo Call* (2600). Diretamente do aparelho celular registrado na rede, deve-se ligar para o número 2600. Ao fazer isso, todo o áudio que o *Asterisk* recebe é retornado ao remetente, ou seja, ao falar no microfone do aparelho celular, pouco tempo depois, ouve-se a fala no fone de ouvido [2].

Essa extensão ajuda a identificar problemas de atraso ou de qualidade do *uplink*. Um pequeno atraso é normal, o cérebro humano lida com atrasos de até 200 milissegundos naturalmente. Porém, diante de atrasos mais longos, uma conversa ficaria extremamente desconfortável [2].

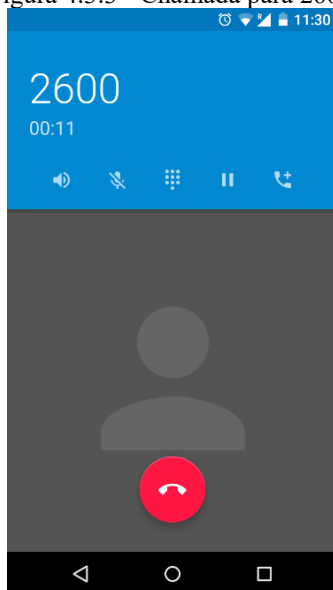
As Figuras 4.3.2 e 4.3.3 mostram chamadas em andamento para as extensões 2602 e 2600 respectivamente. Observa-se que em ambas já foram decorridos 11 segundos. Caso a chamada não fosse completada, assim que colocasse para chamar, seria automaticamente desligado e não haveria nenhuma descrição de tempo.

Figura 4.3.2 - Chamada para 2602.



Fonte: A autora.

Figura 4.3.3 - Chamada para 2600.



Fonte: A autora.

A Figura 4.3.4 mostra as informações geradas sobre as chamadas para as extensões 2602 e 2600 no *software Asterisk*. Observa-se que a chamada para a extensão 2602 teve duração de 27 segundos e a chamada para a extensão 2600 duração de 29 segundos. Além disso, é possível verificar qual MSISDN e IMSI fez cada uma dessas chamadas.

Figura 4.3.4 - Informações das chamadas para as extensões 2602 e 2600 geradas no *Asterisk*.

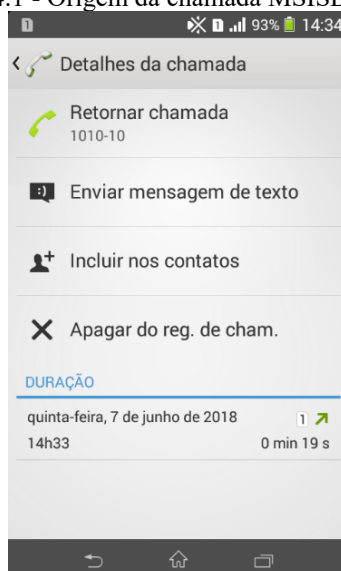
```
[2018-06-06 11:29:16.236] NOTICE[9343][C-00000001]: Ext. h:1 @ default: A-Number=101010 A-Name= A-IMSI=IMSI724340300996638 B-Number=2602 B-Name= B-IMSI= hangupcause=0 dialstatus= hangupdirection=A duration=27 billsec=27
[2018-06-06 11:31:56.360] NOTICE[9611][C-00000003]: Ext. h:1 @ default: A-Number=101010 A-Name= A-IMSI=IMSI724340300996638 B-Number=2600 B-Name= B-IMSI= hangupcause=0 dialstatus= hangupdirection=A duration=29 billsec=29
j*CLI> █
```

Fonte: A autora.

4.4 Teste de Chamadas Entre Dois Aparelhos Celulares

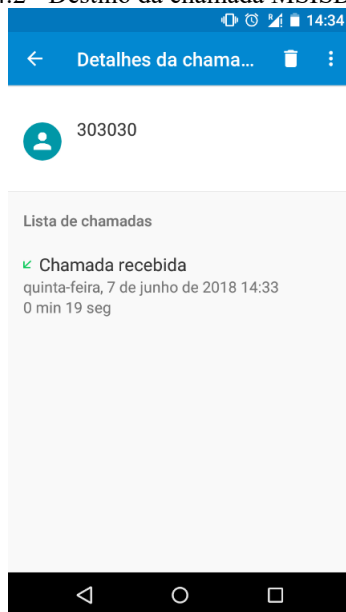
Com os dois aparelhos celulares registrados na rede, é possível fazer chamadas entre eles. Realizou-se uma chamada originada do MSISDN “303030” com destino ao MSISDN “101010”. A Figura 4.4.1 mostra o detalhamento da chamada no aparelho de origem e a Figura 4.4.2 no aparelho de destino. Pode-se observar que o MSISDN de origem aparece corretamente no aparelho de destino.

Figura 4.4.1 - Origem da chamada MSISDN 303030.



Fonte: A autora.

Figura 4.4.2 - Destino da chamada MSISDN 101010.



Fonte: A autora.

4.5 Qualidade do *link*

É possível medir a qualidade do *link* em vez de se basear na percepção do usuário, isso é feito através do comando *chans*. Para utilizá-lo é necessário ter uma chamada ativa [2].

Executa-se o comando a seguir para ver a descrição das informações que são geradas pela execução do comando *chans*.

```
OpenBTS> help chans
```

O comando *chans* fornece inúmeras informações, são elas:

- 1) CN - Número do Canal;
- 2) TN - Número do *Timeslot*;
- 3) *Chan Type* - o tipo de canal dedicado, ou GPRS se reservado para os Serviços de Pacotes;
- 4) *transaction id* - uma ou mais transações da camada 3 em execução neste canal;
- 5) *LAPDm state* – status atual conhecido da mensagem, se houver. Caso contrário, ativo ou inativo;
- 6) *recyc* - *true* se o canal é reciclável, ou seja, pode ser reutilizado agora;

- 7) *Signal* - Intensidade do sinal de *uplink* se o aparelho celular estivesse com potência de transmissão total, em relação ao *GSM.Radio.RSSITarget* configurado. 0 ou menor é ruim;
- 8) *RSSP* - RSSI de *uplink* possível se o aparelho celular estiver com potência total de transmissão;
- 9) *RSSI* - Nível do sinal de *uplink* em dB medido pela estação rádio base, deve estar acima do parâmetro de configuração *GSM.Radio.RSSITarget*;
- 10) *SNR* - Relação Sinal-Ruído no *uplink* medida pela estação rádio base, quanto maior é melhor, menos de 10 é provavelmente inutilizável;
- 11) *FER* - Taxa de perda de *frame* de voz no *uplink*, medida em porcentagem pela estação rádio base;
- 12) *BER* - Taxa de erro de bit no *uplink* antes da decodificação, medida pela estação rádio base em porcentagem;
- 13) *TA* - Avanço de tempo em períodos de símbolo medidos pela estação rádio base;
- 14) *TXPWR* - Potência de transmissão no *uplink* em dBm reportada pelo aparelho celular;
- 15) *TA_DL* - Avanço de tempo em períodos de símbolo relatados pelo aparelho celular;
- 16) *RXLEV_DL* - Nível de sinal no *downlink*, em dBm, reportado pelo aparelho celular;
- 17) *BER_DL* - Taxa de erro de bit no *downlink*, em porcentagem, informada pelo aparelho celular;
- 18) *IMSI* - Valor do IMSI do chip do aparelho celular neste canal, reportado apenas se conhecido;
- 19) *Time* - Duração da conexão no canal em segundos;
- 20) *Timers* - Temporizadores do canal interno;
- 21) *Frames* - Número de *frames* ruins, perdidos e totais enviados apenas para canais de tráfego;
- 22) *Neighbor ARFCN and dBm* - O canal do melhor vizinho e o RSSI de *downlink* reportado pelo aparelho celular.

De acordo com o referencial teórico [2], a Relação Sinal-Ruído (SNR) do *uplink* melhora à medida que o aparelho celular é afastado do rádio. Isso só ocorre porque conforme a distância aumenta, o aparelho celular passa a utilizar mais potência para transmitir seu sinal para a estação rádio base.

Já a potência de transmissão no *uplink* reportada pelo aparelho celular (TXPWR), aumenta à medida que o aparelho celular é afastado do rádio. Isso significa que o aparelho celular utiliza mais energia para transmitir seu sinal para a estação rádio base, e, conseqüentemente, a SNR será maior. Isso ocorre porque a rede instrui de forma independente os aparelhos celulares a transmitirem com diferentes níveis de energia, de modo que todos os sinais recebidos na estação rádio base tenham a mesma energia, o que facilita o processo de demodulação [2].

Já o nível do sinal no *downlink* (RXLEV_DL) reportado pelo aparelho celular diminui à medida que o aparelho celular é afastado do rádio. Isso significa que o aparelho celular recebe o sinal de *downlink* com menos intensidade porque está mais distante [2].

4.5.1 Qualidade do *link* em chamadas para a extensão *Test Tone Call* (2602)

Inicialmente, com todas as chaves da categoria GSM.Radio configuradas nos valores padrões do *OpenBTS*, como mostra a Figura 4.5.1.1, realizou-se chamadas para a extensão *Test Tone Call*, número 2602.

Figura 4.5.1.1 - Valores padrões para chaves da categoria GSM.Radio

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

Fonte: A autora.

As chaves mostradas na Figura 4.5.1.1 com seus respectivos valores padrões, que posteriormente foram alterados para realizar outro teste são mostradas no Quadro 4.5.1.1.

Quadro 4.5.1.1 - Chaves da categoria GSM.Radio

Chave	Valor (Padrão)
<i>GSM.Radio.MaxExpectedDelaySpread</i>	4 períodos de símbolo
<i>GSM.Radio.PowerManager.MaxAttenDB</i>	10 dB
<i>GSM.Radio.PowerManager.MinAttenDB</i>	10 dB
<i>GSM.Radio.RSSITarget</i>	-50 dB
<i>GSM.Radio.SNRTarget</i>	10 dB

Fonte: A autora.

Todos os testes dessa subseção foram realizados com o aparelho celular colocado distante 2 metros, em linha reta, do USRP N210, e após alguns segundos executou-se o comando *chans*.

Posteriormente, foi-se incrementando a distância entre o aparelho celular e o USRP N210 em passos de 1 metro até completar 6 metros. O comando *chans* é executado a cada incremento. Portanto, cinco amostras são geradas em cada teste.

O Quadro 4.5.1.2 mostra os aparelhos celulares, o IMSI dos cartões SIM utilizados nesses aparelhos e o MSISDN dos usuários que foram empregados no teste de medição da qualidade do *link* em chamadas para a extensão 2602.

Quadro 4.5.1.2 - Aparelhos celulares utilizados no teste de qualidade do *link*

Aparelho Celular	IMSI	MSISDN
<i>Motorola Moto G</i>	724340300996638	101010
<i>Sony Xperia E1 D2114</i>	724312921020978	303030
<i>Sony Xperia E1 D2114</i>	724312926827553	404040
<i>Samsung Galaxy Gran Prime G530BT</i>	724340303817952	505050

Fonte: A autora.

Os resultados dos testes são mostrados nas Figuras 4.5.1.2, 4.5.1.3, 4.5.1.4 e 4.5.1.6.

Figura 4.5.1.2 - Teste para aparelho celular *Motorola Moto G* com chip de IMSI 724340300996638.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T156      31      93.2 0.27 -1.0 5 -57 0.00 0:51 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T156      37      90.4 0.20 -0.9 5 -61 0.00 1:55 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T156      34      90.8 0.19 -1.0 5 -68 0.00 2:49 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T156      25      88.8 0.01 -1.0 7 -72 4.53 4:8 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T156      29      89.4 0.32 -0.9 5 -69 2.26 5:14 724340300996638

```

Fonte: A autora.

Figura 4.5.1.3 - Teste para aparelho celular *Sony Xperia E1 D2114* com chip de IMSI 724312921020978.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T157      31      67.1 0.13 0.2 5 -58 0.00 0:53 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T157      38      69.4 0.04 0.1 5 -63 0.00 1:42 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T157      34      68.7 0.10 0.1 5 -70 0.00 2:32 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T157      27      72.6 0.23 0.1 7 -69 2.26 3:13 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T157      28      66.3 0.34 0.2 5 -70 1.13 4:15 724312921020978

```

Fonte: A autora.

Figura 4.5.1.4 - Teste para aparelho celular *Sony Xperia E1 D2114* com chip de IMSI 724312926827553.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T177      25    69.5 0.31 -0.9 11 -58      0.00  0:15 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T177      39    64.5 0.01 -1.1 5 -62      0.00  0:43 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T177      25    72.2 0.36 -1.0 7 -67      0.00  1:14 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T177      28    69  0.08 -0.8 5 -72      2.26  1:56 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T177      35    69.2 0.33 -0.9 5 -68      0.00  2:26 724312926827553

```

Fonte: A autora.

Figura 4.5.1.5 - Teste para aparelho celular *Samsung Galaxy Gran Prime G530BT* com chip de IMSI 724340303817952.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T162      38    73.7 0.20 -0.9 5 -57      0.00  0:32 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T162      38    66.7 0.14 -1.2 5 -62      0.00  1:18 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T162      34    63.6 0.00 -1.1 5 -70      0.00  1:47 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T162      31    66.9 0.10 -1.0 5 -67      0.00  2:21 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3 TCH/F T162      29    63.6 0.54 -1.0 5 -68      1.13  2:59 724340303817952

```

Fonte: A autora.

Em todas as Figuras 4.5.1.2, 4.5.1.3, 4.5.1.4 e 4.5.1.5; conforme evidenciado pela coluna CN (*Channel Number*), há apenas um canal ativo, o canal de número zero, e a coluna TN (Número do *Timeslot*) mostra que foi utilizado o *timeslot* três.

Os valores de SNR, TXPWR, RXLEV_DL, BER_DL e FER de acordo com a distância entre o aparelho celular e o rádio, mostrados nas Figuras 4.5.1.2, 4.5.1.3, 4.5.1.4 e 4.5.1.5 foram transcritos para os Quadros 4.5.1.3, 4.5.1.4, 4.5.1.5, 4.5.1.6 e 4.5.1.7, respectivamente.

Quadro 4.5.1.3 - Valores da Relação Sinal-Ruído (SNR).

Distância entre aparelho celular e o USRP N210 (m)	SNR (dB)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	93,2	67,1	69,5	73,7
3	90,4	69,4	64,5	66,7
4	90,8	68,7	72,2	63,6
5	88,8	72,6	69,0	66,9
6	89,4	66,3	69,2	63,9

Fonte: A autora.

De acordo com os dados do Quadro 4.5.1.3, para todos os aparelhos celulares, observa-se que houve oscilações nos valores da SNR conforme a distância entre os aparelhos celulares e o USRP N210. Isso contradiz com a teoria de [2], que diz que a SNR melhora à medida que o aparelho celular é afastado do equipamento de rádio. Entretanto, observa-se que não houve um aumento considerável nos valores da SNR com o aumento da distância entre o USRP N210 e os aparelhos celulares, isso é devido ao fato de que a potência de transmissão no *uplink* (mostrada no Quadro 4.5.1.4) também não aumentou.

Quadro 4.5.1.4 - Valores da potência de transmissão no *uplink* (TXPWR).

Distância entre aparelho celular e o USRP N210 (m)	TXPWR (dBm)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	5	5	11	5
3	5	5	5	5
4	5	5	7	5
5	7	7	5	5
6	5	5	5	5

Fonte: A autora.

Observa-se no Quadro 4.5.1.4 que a potência de transmissão no *uplink* comportou-se de acordo com a teoria vista em [2] para os aparelhos celulares *Motorola Moto G* (IMSI 724340300996638) e *Sony Xperia E1 D2114* (IMSI 724312921020978) até 5 metros distante do USRP N210. Para o aparelho celular *Samsung Galaxy Gran Prime G530BT* (IMSI 724340303817952) o valor de TXPWR manteve-se constante. Já para o aparelho celular *Sony Xperia E1 D2114* (IMSI 724312926827553) houve oscilações nos valores de TXPWR.

A mínima potência de transmissão no *uplink* foi de 5 dB e a máxima foi de 11 dB.

Quadro 4.5.1.5 - Valores do nível do sinal no *downlink* (RXLEV_DL).

Distância entre aparelho celular e o USRP N210 (m)	RXLEV_DL (dBm)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	-57	-58	-58	-57
3	-61	-63	-62	-62
4	-68	-70	-67	-70
5	-72	-69	-72	-67
6	-69	-70	-68	-68

Fonte: A autora.

De acordo com os dados do Quadro 4.5.1.5, em ambos os aparelhos celulares, o nível do sinal no *downlink* (RXLEV_DL) diminuiu à medida que esses foram afastados até 4 metros do USRP N210, o que condiz com a teoria vista em [2].

Quadro 4.5.1.6 - Valores da taxa de erro de bit no *downlink* (BER_DL).

Distância entre aparelho celular e o USRP N210 (m)	BER_DL (%)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	0,00	0,00	0,00	0,00
3	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00
5	4,53	2,26	2,26	0,00
6	2,26	1,13	0,00	1,13

Fonte: A autora.

De acordo com os dados do Quadro 4.5.1.6, para todos os aparelhos celulares não houve erro de bit no *downlink* até a distância de 3 metros do USRP N210.

Quadro 4.5.1.7 – Valores da taxa de perda de frame de voz no *uplink* (FER).

Distância entre aparelho celular e o USRP N210 (m)	FER (%)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	0,27	0,13	0,31	0,20
3	0,20	0,04	0,01	0,14
4	0,19	0,10	0,36	0,00
5	0,01	0,23	0,08	0,10
6	0,32	0,34	0,33	0,54

Fonte: A autora.

Observa-se no Quadro 4.5.1.7 que o valor máximo da taxa de perda de *frame* de voz no *uplink* (FER) foi de 0,54 %, e o valor mínimo foi de 0,01 %.

Realizou-se outro teste em que as chaves mostradas no Quadro 4.5.1.8 foram configuradas para que a qualidade do *link* nas chamadas para extensão 2602 tivessem um desempenho melhor. A configuração dessas chaves é mostrada na Figura 4.5.1.6.

Quadro 4.5.1.8 - Chaves da categoria GSM.Radio configuradas para um desempenho melhor.

Chave	Valor (Padrão)
<i>GSM.Radio.MaxExpectedDelaySpread</i>	1 período de símbolo
<i>GSM.Radio.PowerManager.MaxAttenDB</i>	0 dB
<i>GSM.Radio.PowerManager.MinAttenDB</i>	0 dB
<i>GSM.Radio.RSSITarget</i>	-25 dB
<i>GSM.Radio.SNRTarget</i>	20 dB

Fonte: A autora.

Figura 4.5.1.6 - Chaves da categoria GSM.Radio configuradas para melhor desempenho.

```
OpenBTS> config GSM.Radio
GSM.Radio.ARFCNs 1      [default]
GSM.Radio.Band 900      [default]
GSM.Radio.C0 51        [default]
GSM.Radio.MaxExpectedDelaySpread 1
GSM.Radio.PowerManager.MaxAttenDB 0
GSM.Radio.PowerManager.MinAttenDB 0      [default]
GSM.Radio.RSSITarget -25
GSM.Radio.SNRTarget 20
```

Fonte: A autora.

Os resultados dos testes utilizando configurações para obter um melhor desempenho são mostrados nas Figuras 4.5.1.7, 4.5.1.8, 4.5.1.9 e 4.5.1.10.

A variação dos valores da SNR, TXPWR, RXLEV_DL, BER_DL e FER de acordo com a distância entre o aparelho celular e o rádio, mostrados nas Figuras 4.5.1.7, 4.5.1.8, 4.5.1.9 e 4.5.1.10 foram transcritas para os Quadros 4.5.1.9, 4.5.1.10, 4.5.1.11, 4.5.1.12 e 4.5.1.13, respectivamente.

Figura 4.5.1.7 - Teste para aparelho celular *Motorola Moto G* com chip de IMSI 724340300996638.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T185      9      70.9 0.15 -0.8 23 -48      0.00 0:52 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T185     10      78.1 0.44 -1.0 23 -51      0.00 1:23 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T185      9     91.4 0.27 -0.9 25 -60      0.00 1:52 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T185      5      71  0.06 -0.9 29 -59      0.00 2:34 724340300996638

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T185     10     73.7 0.21 -0.9 23 -60      0.00 3:6 724340300996638

```

Fonte: A autora.

Figura 4.5.1.8 - Teste para aparelho celular *Sony Xperia E1 D2114* com chip de IMSI 724312921020978.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T190      4     71.7 0.07 -1.0 31 -50      0.00 0:21 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T190      7     74.2 0.09 -0.9 25 -58      0.00 0:47 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T190      8     70.9 0.21 -1.0 25 -63      0.00 1:9 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T190      6     67.6 0.07 -1.0 27 -58      0.00 1:45 724312921020978

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id          dB      pct sym dBm dBm      pct
0  3 TCH/F T190      4     73.2 0.42 -1.0 29 -63      0.00 2:22 724312921020978

```

Fonte: A autora.

Figura 4.5.1.9 - Teste para aparelho celular *Sony Xperia E1 D2114* com chip de IMSI 724312926827553.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T223      3      73.3 0.00 -1.0 27 -48      0.00 0:18 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T223     10      73.7 0.00 -0.9 21 -53      0.00 0:54 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T223      1      69.2 0.00 -0.9 31 -57      0.00 1:32 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T223     -1      73.1 0.00 -0.9 33 -61      0.00 2:12 724312926827553

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T223      4      77.9 0.00 -0.9 29 -60      0.00 2:52 724312926827553

```

Fonte: A autora.

Figura 4.5.1.10 - Teste para aparelho celular *Samsung Galaxy Gran Prime G530BT* com chip de IMSI 724340303817952.

```

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T199     11      71.4 0.12 -0.8 23 -48      0.00 0:34 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T199      6      63.1 0.38 -1.1 27 -53      0.00 1:14 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T199      6      75.9 0.47 -1.1 27 -66      0.00 1:48 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T199     -3      69.8 0.14 -1.1 33 -59      0.00 2:30 724340303817952

OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0  3  TCH/F T199      4      68.9 0.16 -1.1 29 -57      0.00 3:14 724340303817952

```

Fonte: A autora.

Quadro 4.5.1.9 - Valores da Relação Sinal-Ruído (SNR).

Distância entre aparelho celular e o USRP N210 (m)	SNR (dB)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	70,9	71,7	73,3	71,4
3	78,1	74,2	73,7	63,1
4	91,4	70,9	69,2	75,9
5	71	67,6	73,1	69,8
6	73,7	73,2	77,9	68,9

Fonte: A autora.

Observa-se no Quadro 4.5.1.9 que assim como no teste realizado com as configurações padrões das chaves da categoria *GSM.Radio*, os resultados da SNR apresentaram oscilações conforme a distância entre os aparelhos celulares e o USRP N210, o que contradiz a teoria de [2].

Quadro 4.5.1.10 - Valores da potência de transmissão no *uplink* (TXPWR).

Distância entre aparelho celular e o USRP N210 (m)	TXPWR (dBm)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	23	31	27	23
3	23	25	21	27
4	25	25	31	27
5	29	27	33	33
6	23	29	29	29

Fonte: A autora.

Enquanto a mínima potência de transmissão no *uplink* foi de 5 dB e a máxima foi de 11 dB para as chaves da categoria *GSM.Radio* configuradas nos valores padrões, o Quadro 4.5.1.10 mostra que a mínima potência de transmissão foi de 21 dB e a máxima de 33 dB para as chaves da categoria *GSM.Radio* configuradas para obter um melhor desempenho.

Esse aumento nos valores da potência de transmissão é justificado pela alteração nos valores das chaves *GSM.Radio.RSSITarget* e *GSM.Radio.SNRTarget*, pois como mostrado na subseção 3.2 (Configurações Básicas do *OpenBTS*), o *loop* de controle de potência do aparelho celular ajusta a potência de transmissão para satisfazer os valores dessas chaves.

Quadro 4.5.1.11 - Valores do nível do sinal no *downlink* (RXLEV_DL).

Distância entre aparelho celular e o USRP N210 (m)	RXLEV_DL (dBm)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	-48	-50	-48	-48
3	-51	-58	-53	-53
4	-60	-63	-57	-66
5	-59	-58	-61	-59
6	-60	-63	-60	-57

Fonte: A autora.

Assim como nos resultados do nível do sinal no *downlink* (RXLEV_DL) obtidos com as chaves da categoria *GSM.Radio* configuradas nos valores padrões, o Quadro 4.5.1.11 mostra que em todos os aparelhos celulares, o RXLEV_DL diminuiu à medida que esses foram afastados até 4 metros do USRP N210, o que condiz com a teoria vista em [2].

Quadro 4.5.1.12 - Valores da taxa de erro de bit no *downlink* (BER_DL).

Distância entre aparelho celular e o USRP N210 (m)	BER_DL (%)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	0,00	0,00	0,00	0,00
3	0,00	0,00	0,00	0,00
4	0,00	0,00	0,00	0,00
5	0,00	0,00	0,00	0,00
6	0,00	0,00	0,00	0,00

Fonte: A autora.

Observa-se no Quadro 4.5.1.12 que o resultado do teste para a taxa de erro de bit no *downlink* (BER_DL) utilizando configurações para obter um melhor desempenho foram satisfatórios, pois a BER_DL foi de 0,00% para ambos os aparelhos celulares, o que não ocorreu no teste utilizando os valores padrões das chaves da categoria *GSM.Radio*. Isso pode ser justificado pela alteração das chaves *GSM.Radio.PowerManager.MaxAttenDB* e *GSM.Radio.PowerManager.MinAttenDB*. Essas chaves foram configuradas para 0 dB, ou seja, para não haver atenuação na potência de transmissão do *downlink*.

Quadro 4.5.1.13 - Valores da taxa de perda de frame de voz no *uplink* (FER).

Distância entre aparelho celular e o USRP N210 (m)	FER (%)			
	<i>Motorola Moto G</i>	<i>Sony Xperia E1 D2114</i>	<i>Sony Xperia E1 D2114</i>	<i>Samsung Galaxy Gran Prime G530BT</i>
	IMSI 724340300996638	IMSI 724312921020978	IMSI 724312926827553	IMSI 724340303817952
2	0,15	0,07	0,00	0,12
3	0,44	0,09	0,00	0,38
4	0,27	0,21	0,00	0,47
5	0,06	0,07	0,00	0,14
6	0,21	0,42	0,00	0,16

Fonte: A autora.

Observa-se no Quadro 4.5.1.13 que o valor máximo da taxa de perda de *frame* de voz no *uplink* (FER) foi de 0,47 %, e o valor mínimo foi de 0,00 %, isso para as chaves da categoria *GSM.Radio* configuradas para obter um melhor desempenho na qualidade do *link*. Já para essas mesmas chaves configuradas nos valores padrões, o valor máximo da FER foi de 0,54 %, e o valor mínimo foi de 0,01 %.

4.5.2 Qualidade do *link* em chamadas entre dois aparelhos celulares

Todos os testes dessa seção foram realizados com as chaves da categoria *GSM.Radio* configuradas para obter um melhor desempenho na qualidade do *link* das chamadas. Os valores de configuração para essas chaves são mostrados na Figura 4.5.1.6 da subseção 4.5.1.

Inicialmente, realizou-se uma chamada originada do aparelho celular *Motorola Moto G*, MSISDN 101010 e IMSI 724340300996638, colocado a 2 metros de distância do USRP N210 com destino ao aparelho celular *Sony Xperia E1 D2114*, MSISDN 404040 e IMSI 724312926827553, colocado a 6 metros de distância do USRP N210. Executou-se o comando *chans* e os dados obtidos são mostrados na Figura 4.5.2.1.

Figura 4.5.2.1 - Medindo qualidade do *link* para chamada entre MSISDN 101010 e MSISDN 404040.

```
OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0 3 TCH/F T217    10    73.7 0.37 -0.9 23 -48    0.00 3:3 724340300996638
0 4 TCH/F T218    10    69.9 0.00 -0.9 23 -56    0.00 2:57 724312926827553
```

Fonte: A autora.

Posteriormente, realizou-se uma chamada originada do aparelho celular *Motorola Moto G*, MSISDN 101010 e IMSI 724340300996638, colocado a 2 metros de distância do USRP N210 com destino ao aparelho celular *Samsung Galaxy Gran Prime G530BT*, MSISDN

505050 e IMSI 724340303817952, colocado a 6 metros de distância do USRP N210. Executou-se o comando *chans* e os dados obtidos são mostrados na Figura 4.5.2.2.

Figura 4.5.2.2 - Medindo qualidade do *link* para chamada entre MSISDN 101010 e MSISDN 505050.

```
OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0 3 TCH/F T200    6    89.3 0.40 -0.9 27 -48    0.00 2:53 724340300996638
0 4 TCH/F T201   10    68.1 0.00 -1.0 23 -58    0.00 2:47 724340303817952
```

Fonte: A autora.

Por último, realizou-se uma chamada originada do aparelho celular *Samsung Galaxy Gran Prime G530BT*, MSISDN 505050 e IMSI 724340303817952, colocado a 2 metros de distância do USRP N210 com destino ao aparelho celular *Sony Xperia EI D2114*, MSISDN 404040 e IMSI 724312926827553, colocado a 6 metros de distância do USRP N210. Executou-se o comando *chans* e os dados obtidos são mostrados na Figura 4.5.2.3.

Figura 4.5.2.3 - Medindo qualidade do *link* para chamada entre MSISDN 505050 e MSISDN 404040.

```
OpenBTS> chans
CN TN chan transaction Signal SNR FER TA TXPWR RXLEV_DL BER_DL Time IMSI
   type id      dB      pct sym dBm dBm      pct
0 3 TCH/F T221   13    65.9 0.04 -1.2 21 -48    0.00 2:44 724340303817952
0 4 TCH/F T222    9    69.2 0.00 -1.0 23 -57    0.00 2:35 724312926827553
```

Fonte: A autora.

Os valores de CN, TN, SNR, FER, TXPWR, RXLEV_DL, BER_DL e IMSI mostrados nas Figuras 4.5.2.1, 4.5.2.2 e 4.5.2.3 foram transcritos para o Quadro 4.5.2.1.

Quadro 4.5.2.1 - Testes de Qualidade do *link* em chamadas entre dois aparelhos celulares.

	CN	TN	SNR dB	FER %	TXPWR dBm	RXLEV_DL dBm	BER_DL %	IMSI
Origem MSISDN 101010	0	3	73,7	0,37	23	-48	0,00	724340300996638
Destino MSISDN 404040	0	4	69,9	0,00	23	-56	0,00	724312926827553
Origem MSISDN 101010	0	3	89,3	0,40	27	-48	0,00	724340300996638
Destino MSISDN 505050	0	4	68,1	0,00	23	-58	0,00	724340303817952
Origem MSISDN 505050	0	3	65,9	0,04	21	-48	0,00	724340303817952
Destino MSISDN 404040	0	4	69,2	0,00	23	-57	0,00	724312926827553

Fonte: A autora.

Com base nos dados mostrados no Quadro 4.5.2.1, observa-se o seguinte:

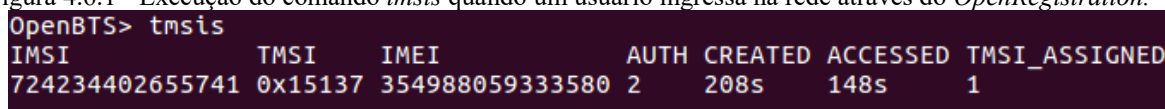
- 1) A coluna CN mostra que em ambas as chamadas foi utilizado o canal de número zero;

- 2) A coluna TN mostra que foi utilizado o *time slot* três para os MSISDNs que originaram as chamadas e o *time slot* quatro para os MSISDNs aos quais as chamadas foram destinadas;
- 3) A coluna FER mostra que todos os MSISDNs que originaram as chamadas tiveram perda de *frame* de voz no *uplink*, o que não ocorreu para os MSISDNs aos quais as chamadas foram destinadas;
- 4) A coluna RXLEV_DL mostra que os MSISDNs aos quais as chamadas foram destinadas obtiveram menores taxas do sinal de *downlink*, o que é explicado pelo fato deles estarem mais distantes do USRP N210 (6 metros).
- 5) A coluna BER_DL mostra que a taxa de erro de bit no *downlink* foi de 0,00 % para todas as chamadas.

4.6 Teste do Serviço *OpenRegistration*

Para verificar o correto funcionamento do serviço *OpenRegistration*, utilizou-se o aparelho celular *Motorola Moto G* com um cartão SIM não cadastrado na rede. Então, selecionou-se a rede JRTelecom na lista de redes disponíveis. Posteriormente, executou-se o comando “*tmsis*” assim como mostra a Figura 4.6.1.

Figura 4.6.1 - Execução do comando *tmsis* quando um usuário ingressa na rede através do *OpenRegistration*.

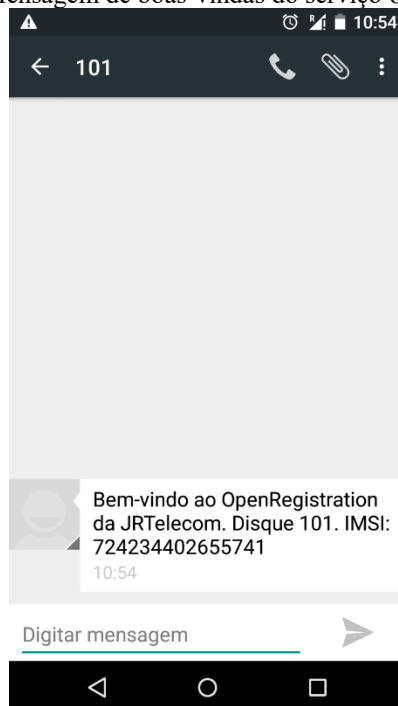


OpenBTS> tmsis						
IMSI	TMSI	IMEI	AUTH	CREATED	ACCESSED	TMSI_ASSIGNED
724234402655741	0x15137	354988059333580	2	208s	148s	1

Fonte: A autora.

Observa-se na Figura 4.5.1 que a coluna AUTH se encontra preenchida com o número dois. Isso significa que o IMSI 724234402655741 não é um assinante da rede, porém, ingressou-se nela através do serviço *OpenRegistration*.

A Figura 4.6.2 mostra a mensagem de boas-vindas ao serviço *OpenRegistration* exibida no aparelho celular do usuário. Observa-se que nessa mensagem contém o IMSI do cartão SIM que o usuário utilizou para entrar na rede.

Figura 4.6.2 - Mensagem de boas-vindas do serviço *OpenRegistration*.

Fonte: A autora.

Na subseção 3.4, o número 101 foi atribuído ao usuário “Assistência”. O usuário que ingressa na rede através do *OpenRegistration* é instruído a buscar ajuda ligando para esse número.

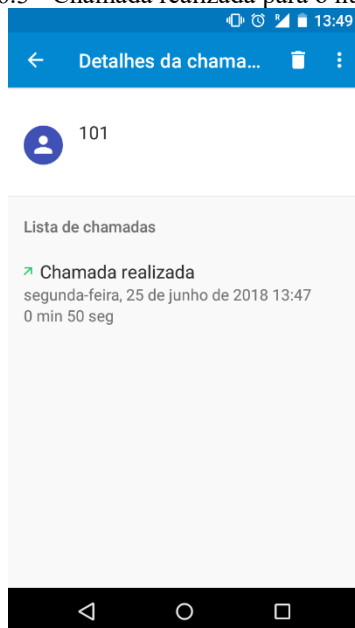
A Figura 4.6.3 mostra os detalhes da chamada para o número 101 no aparelho celular do usuário do *OpenRegistration*. A Figura 4.6.4 mostra os detalhes da chamada sendo recebida no aparelho celular do usuário “Assistência”.

Observa-se que na Figura 4.6.4 a chamada é recebida de um número desconhecido. Isso é porque ainda não tem nenhum MSISDN atribuído ao usuário do *OpenRegistration*. No entanto, durante uma conversa, o usuário “Assistência” pode pedir ao usuário do *OpenRegistration* para enviar via SMS um MSISDN que contém dez dígitos. Assim, o usuário “Assistência” cadastrará o usuário do *OpenRegistration* na rede.

Um usuário do *OpenRegistration* não cadastrado na rede pode fazer ligações para o número 101 e para qualquer assinante da rede. No entanto, o único SMS que pode enviar é o SMS que requisita um MSISDN.

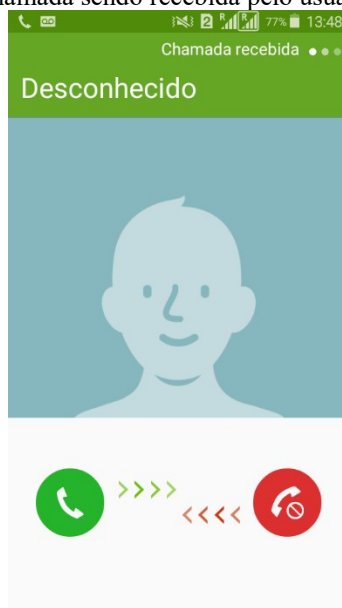
Quando um usuário do *OpenRegistration* não cadastrado na rede realizar ligações para um que seja cadastrado, o número de quem origina a chamada aparecerá como desconhecido no aparelho celular ao qual a chamada foi destinada.

Figura 4.6.3 - Chamada realizada para o número 101.



Fonte: A autora.

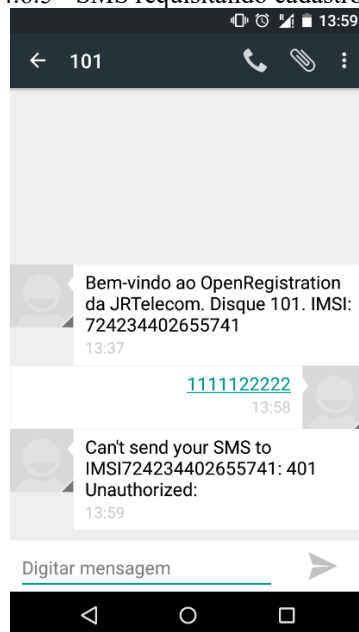
Figura 4.6.4 - Chamada sendo recebida pelo usuário “Assistência”.



Fonte: A autora.

Posteriormente, o usuário do *OpenRegistration* envia por SMS o MSISDN 1111122222 requisitando que o cadastre na rede (Figura 4.6.5).

Figura 4.6.5 - SMS requisitando cadastro na rede.



Fonte: A autora.

Embora o usuário do *OpenRegistration* tenha recebido a mensagem “*Can’t send your SMS to IMSI 724234402655741: 401 Unauthorized:*”, esse usuário foi cadastrado com sucesso. Isso é confirmado ao executar o comando *tmsis* como mostra a Figura 4.6.6.

Observa-se na Figura 4.6.6 que a coluna AUTH se encontra preenchida com o número um. Isso significa que o IMSI 724234402655741 passou a ser um assinante da rede.

Figura 4.6.6 - Execução do comando *tmsis* após o usuário do *OpenRegistration* ser cadastrado na rede.

```
OpenBTS> tmsis
IMSI      TMSI      IMEI      AUTH  CREATED  ACCESSED  TMSI_ASSIGNED
724234402655741 0x5acc4 354988059333580 1      27s      25s      1
```

Fonte: A autora.

Além do uso do comando *tmsis*, para confirmar o cadastro do usuário do *OpenRegistration* na rede, basta acessar a rota *cd /dev/NodeManager* e executar o comando *./nmcli.py sipauthserve read* para visualizar todos os assinantes cadastrados na rede (Figura 4.6.7).

Figura 4.6.7 - Usuário do *OpenRegistration* presente na lista de assinantes cadastrados na rede.

```
openbts@j:~/dev/NodeManager$ ./nmcli.py sipauthserve subscribers read
raw request: {"command":"subscribers","action":"read","key":"","value":""}
raw response: {
  "code" : 200,
  "data" : [
    {
      "imsi" : "IMSI724340300996638",
      "msisdn" : "101010",
      "name" : "Juliana"
    },
    {
      "imsi" : "IMSI724340302312523",
      "msisdn" : "202020",
      "name" : "Gustavo"
    },
    {
      "imsi" : "IMSI724312921020978",
      "msisdn" : "303030",
      "name" : "Sony1"
    },
    {
      "imsi" : "IMSI724312926827553",
      "msisdn" : "404040",
      "name" : "Sony2"
    },
    {
      "imsi" : "IMSI724340303817952",
      "msisdn" : "505050",
      "name" : "Samsung1"
    },
    {
      "imsi" : "IMSI724109393365788",
      "msisdn" : "101",
      "name" : "Assistência"
    },
    {
      "imsi" : "IMSI724234402655741",
      "msisdn" : "1111122222",
      "name" : "IMSI724234402655741"
    }
  ]
}
```

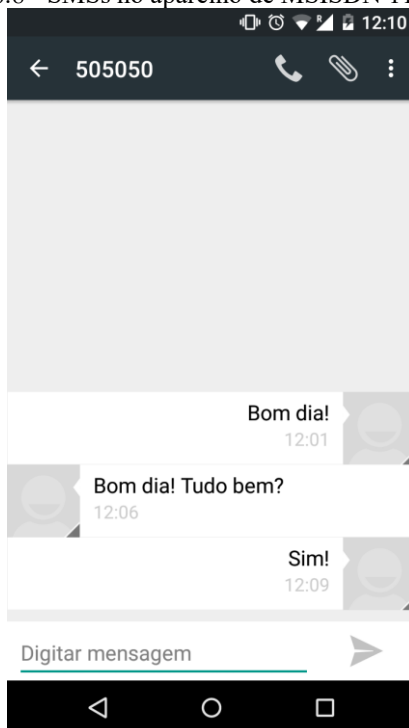
Fonte: A autora.

Observa-se na Figura 4.6.7 que os dados do último usuário da lista correspondem aos dados do usuário do *OpenRegistration*: IMSI do cartão SIM utilizado (724234402655741) e o MSISDN enviado via SMS (1111122222). Já o campo “name” é preenchido com o próprio IMSI.

Após o usuário do *OpenRegistration* ser cadastrado na rede, ele passa a ser um usuário da rede JRTelecom, e pode enviar mensagens SMS para qualquer assinante dessa rede. Para verificar isso, trocou-se SMSs entre o MSISDN 1111122222 e o MSISDN 505050. A Figura 4.6.8 mostra os SMSs no aparelho de MSISDN 1111122222 e a Figura 4.6.9 os SMSs no aparelho de MSISDN 505050.

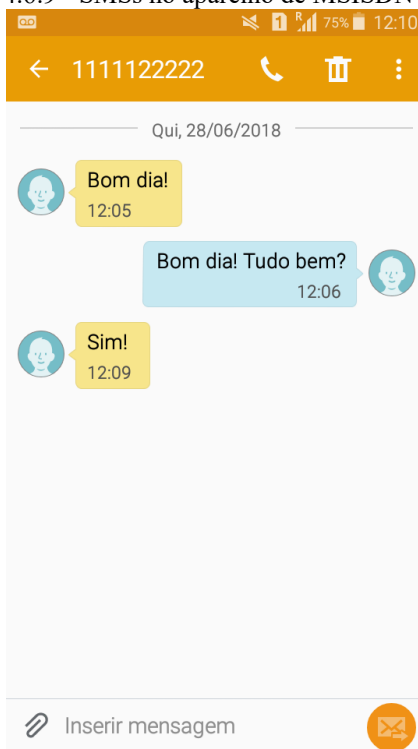
Além disso, nas ligações o MSISDN 1111122222 passará a ser mostrado no aparelho celular ao qual a chamada foi destinada. A Figura 4.6.10 mostra uma chamada sendo recebida pelo MSISDN 505050 originada do MSISDN 1111122222.

Figura 4.6.8 - SMSs no aparelho de MSISDN 1111122222.



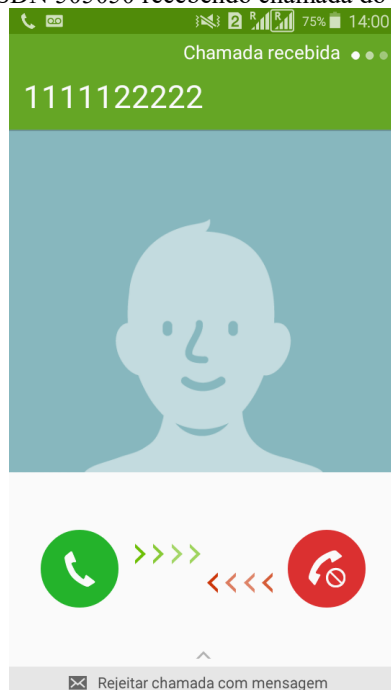
Fonte: A autora.

Figura 4.6.9 - SMSs no aparelho de MSISDN 505050.



Fonte: A autora.

Figura 4.6.10 - MSISDN 505050 recebendo chamada do MSIDN 111122222.



Fonte: A autora.

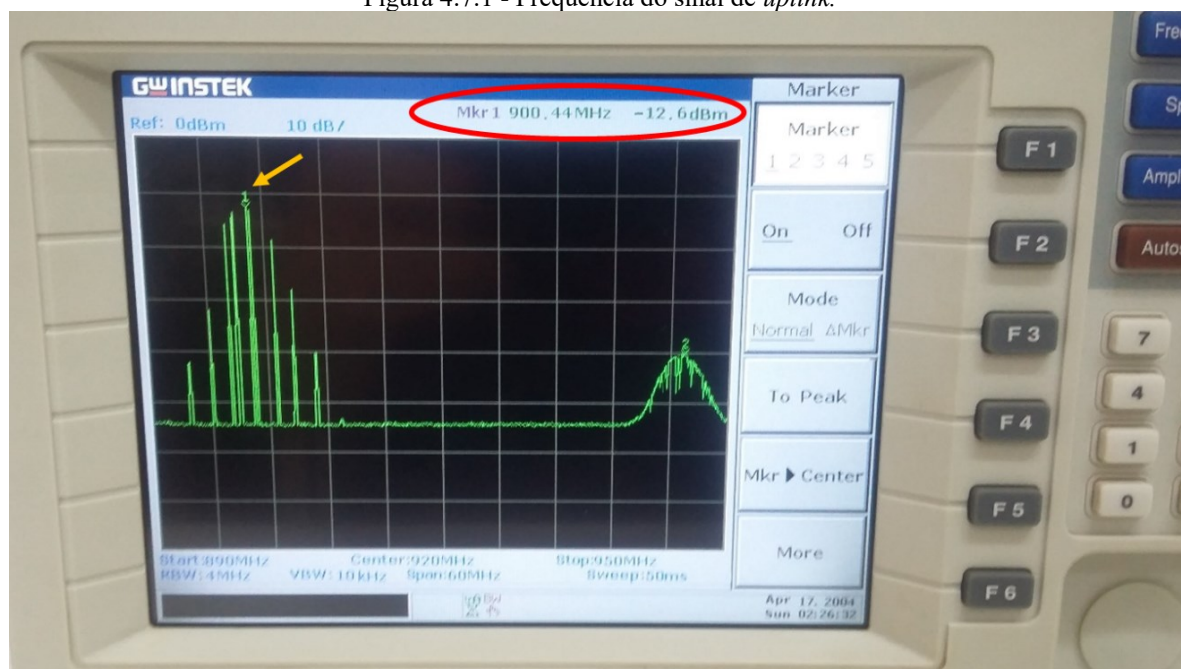
4.7 Verificação das Frequências da Portadora

Conforme descrito na seção 3.2 (Configurações Básicas do *OpenBTS*), o ARFCN #51 utiliza a frequência de 945,401 MHz para o *downlink* e a frequência de 900,401 MHz para o *uplink*. Realizou-se um teste para averiguar essas frequências utilizando um equipamento Analisador de Espectro da marca *GWInstek*, modelo *GSP-830*.

Para isso, diretamente de um aparelho celular registrado na rede, realizou-se uma chamada para a extensão *Test Tone Call* (número 2602), assim apenas um *timeslot* é alocado para o sinal de *uplink*.

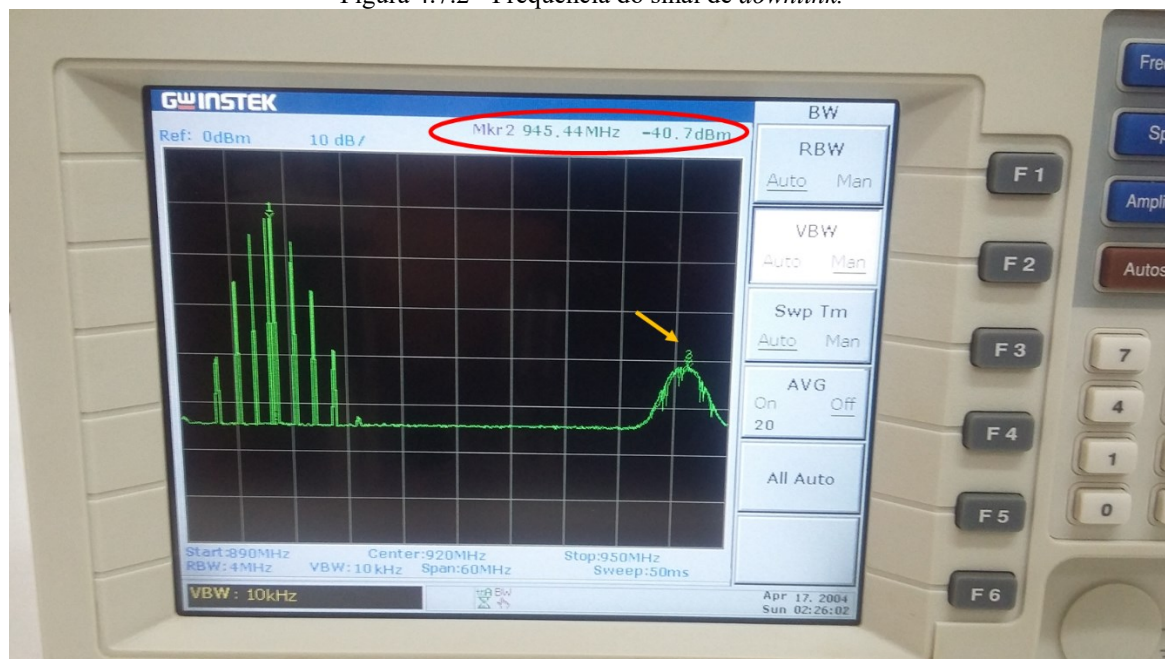
O aparelho celular foi colocado bem próximo à antena do Analisador de Espectro e o USRP N210 foi colocado mais distante, consequentemente a amplitude do sinal de *uplink* será maior que amplitude do sinal de *downlink*. Essa amplitude representa a potência do sinal.

As Figuras 4.7.1 e 4.7.2 mostram, respectivamente a frequência do sinal de *uplink* e de *downlink*.

Figura 4.7.1 - Frequência do sinal de *uplink*.

Fonte: A autora.

Observa-se na Figura 4.7.1 que a frequência para o *uplink* informada pelo Analisador de Espectro é de 900,44 MHz.

Figura 4.7.2 - Frequência do sinal de *downlink*.

Fonte: A autora.

Observa-se na Figura 4.7.2 que a frequência para *downlink* informada pelo Analisador de Espectro é de 945,44 MHz.

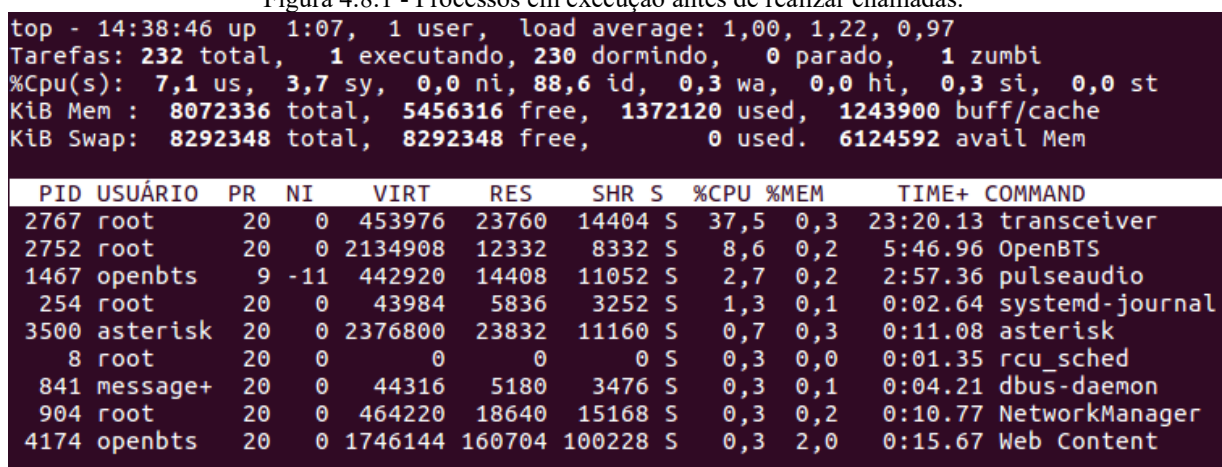
4.8 Análise do Custo Computacional

Como mostrado na subseção 3.6.3 (Distorção do Sinal), o *OpenBTS* analisa um grande número de períodos de símbolo ao tentar cancelar as distorções *multipath*, o que acarreta em um trabalho computacional pesado.

Realizou-se testes utilizando o comando “*top*” do *Linux* para analisar o consumo computacional em chamadas para a extensão *Test Tone Call* (número 2602). O comando “*top -i*” exibe detalhes dos processos que estão sendo executados no sistema.

A princípio executou-se o comando “*top -i*” para listar todos os processos em execução antes de realizar as chamadas, o resultado é mostrado na Figura 4.8.1.

Figura 4.8.1 - Processos em execução antes de realizar chamadas.



```
top - 14:38:46 up 1:07, 1 user, load average: 1,00, 1,22, 0,97
Tarefas: 232 total, 1 executando, 230 dormindo, 0 parado, 1 zumbi
%Cpu(s): 7,1 us, 3,7 sy, 0,0 ni, 88,6 id, 0,3 wa, 0,0 hi, 0,3 si, 0,0 st
KiB Mem : 8072336 total, 5456316 free, 1372120 used, 1243900 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 6124592 avail Mem
```

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	37,5	0,3	23:20.13	transceiver
2752	root	20	0	2134908	12332	8332	S	8,6	0,2	5:46.96	OpenBTS
1467	openbts	9	-11	442920	14408	11052	S	2,7	0,2	2:57.36	pulseaudio
254	root	20	0	43984	5836	3252	S	1,3	0,1	0:02.64	systemd-journal
3500	asterisk	20	0	2376800	23832	11160	S	0,7	0,3	0:11.08	asterisk
8	root	20	0	0	0	0	S	0,3	0,0	0:01.35	rcu_sched
841	message+	20	0	44316	5180	3476	S	0,3	0,1	0:04.21	dbus-daemon
904	root	20	0	464220	18640	15168	S	0,3	0,2	0:10.77	NetworkManager
4174	openbts	20	0	1746144	160704	100228	S	0,3	2,0	0:15.67	Web Content

Fonte: A autora.

Com base nos dados mostrados na Figura 4.8.1, verificou-se o PID (Identificador do Processo) para os processos *transceiver*, *OpenBTS* e *Asterisk*, pois estes são os componentes utilizados durante as chamadas. De posse do PID desses processos, realizou-se os testes com o comando “*top -p2767,2752,3500*” para monitorar apenas os processos de interesse.

Inicialmente, realizou-se testes com a chave *GSM.Radio.MaxExpectedDelaySpread* configurada no valor 4. As Figuras 4.8.2, 4.8.3, 4.8.4 e 4.8.5 mostram o resultado da análise da carga computacional para 1 chamada, 2 chamadas, 3 chamadas e 4 chamadas, respectivamente.

Figura 4.8.2 - Consumo computacional durante uma chamada.

```
openbts@j:~$ top -p2767,2752,3500
```

top - 14:50:22 up 1:19, 1 user, load average: 1,24, 1,36, 1,24
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 12,2 us, 4,7 sy, 0,0 ni, 82,7 id, 0,0 wa, 0,0 hi, 0,4 si, 0,0 st
KiB Mem : 8072336 total, 5104712 free, 1561116 used, 1406508 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5832804 avail Mem

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	35,0	0,3	27:29.26	transceiver
2752	root	20	0	2134908	12712	8392	S	8,7	0,2	7:03.68	OpenBTS
3500	asterisk	20	0	2378288	24716	11160	S	0,3	0,3	0:19.63	asterisk

Fonte: A autora.

Figura 4.8.3 - Consumo computacional durante duas chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

top - 14:51:07 up 1:19, 1 user, load average: 1,03, 1,29, 1,22
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 11,0 us, 4,9 sy, 0,0 ni, 83,3 id, 0,3 wa, 0,0 hi, 0,5 si, 0,0 st
KiB Mem : 8072336 total, 5106144 free, 1560764 used, 1405428 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5834596 avail Mem

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	41,3	0,3	27:46.75	transceiver
2752	root	20	0	2134908	12976	8392	S	12,7	0,2	7:08.40	OpenBTS
3500	asterisk	20	0	2378288	24716	11160	S	1,3	0,3	0:20.05	asterisk

Fonte: A autora.

Figura 4.8.4 - Consumo computacional durante três chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

top - 14:51:34 up 1:20, 1 user, load average: 1,08, 1,28, 1,22
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 9,7 us, 4,6 sy, 0,0 ni, 85,2 id, 0,0 wa, 0,0 hi, 0,5 si, 0,0 st
KiB Mem : 8072336 total, 5107604 free, 1563796 used, 1400936 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5836372 avail Mem

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	43,3	0,3	27:57.75	transceiver
2752	root	20	0	2134908	12976	8392	S	16,0	0,2	7:12.03	OpenBTS
3500	asterisk	20	0	2378288	24716	11160	S	2,7	0,3	0:20.62	asterisk

Fonte: A autora.

Figura 4.8.5 - Consumo computacional durante quatro chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

top - 14:52:25 up 1:21, 1 user, load average: 1,44, 1,35, 1,24
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 15,2 us, 5,8 sy, 0,0 ni, 78,0 id, 0,0 wa, 0,0 hi, 0,9 si, 0,0 st
KiB Mem : 8072336 total, 5107060 free, 1563620 used, 1401656 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5836512 avail Mem

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	46,5	0,3	28:20.13	transceiver
2752	root	20	0	2134908	12976	8392	S	24,9	0,2	7:22.39	OpenBTS
3500	asterisk	20	0	2378288	24716	11160	S	5,3	0,3	0:22.78	asterisk

Fonte: A autora.

Repetiu-se os testes anteriores com a chave *GSM.Radio.MaxExpectedDelaySpread* configurada no valor 1. As Figuras 4.8.6, 4.8.7, 4.8.8 e 4.8.9 mostram o resultado da análise da carga computacional para 1 chamada, 2 chamadas, 3 chamadas e 4 chamadas, respectivamente.

Figura 4.8.6 - Consumo computacional durante uma chamada.

```
openbts@j:~$ top -p2767,2752,3500
```

```
top - 14:45:34 up 1:14, 1 user, load average: 1,15, 1,59, 1,25
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 10,1 us, 3,6 sy, 0,0 ni, 83,6 id, 2,2 wa, 0,0 hi, 0,5 si, 0,0 st
KiB Mem : 8072336 total, 5079596 free, 1613028 used, 1379712 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5805332 avail Mem
```

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	34,0	0,3	25:27.46	transceiver
2752	root	20	0	2134908	12332	8332	S	9,0	0,2	6:19.83	OpenBTS
3500	asterisk	20	0	2376800	23832	11160	S	0,3	0,3	0:12.11	asterisk

Fonte: A autora.

Figura 4.8.7 - Consumo computacional durante duas chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

```
top - 14:46:16 up 1:14, 1 user, load average: 1,10, 1,52, 1,25
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 9,6 us, 4,9 sy, 0,0 ni, 85,2 id, 0,0 wa, 0,0 hi, 0,3 si, 0,0 st
KiB Mem : 8072336 total, 5080544 free, 1613684 used, 1378108 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5806596 avail Mem
```

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	39,3	0,3	25:42.54	transceiver
2752	root	20	0	2134908	12564	8392	S	13,0	0,2	6:24.01	OpenBTS
3500	asterisk	20	0	2376800	23832	11160	S	1,7	0,3	0:12.37	asterisk

Fonte: A autora.

Figura 4.8.8 - Consumo computacional durante três chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

```
top - 14:46:52 up 1:15, 1 user, load average: 0,78, 1,40, 1,22
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 11,8 us, 4,4 sy, 0,0 ni, 82,8 id, 0,6 wa, 0,0 hi, 0,4 si, 0,0 st
KiB Mem : 8072336 total, 5078432 free, 1614188 used, 1379716 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5804788 avail Mem
```

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	45,5	0,3	25:57.41	transceiver
2752	root	20	0	2134908	12564	8392	S	16,6	0,2	6:29.06	OpenBTS
3500	asterisk	20	0	2377296	23832	11160	S	2,7	0,3	0:13.06	asterisk

Fonte: A autora.

Figura 4.8.9 - Consumo computacional durante quatro chamadas simultâneas.

```
openbts@j:~$ top -p2767,2752,3500
```

```
top - 14:47:31 up 1:16, 1 user, load average: 0,81, 1,34, 1,21
Tarefas: 3 total, 0 executando, 3 dormindo, 0 parado, 0 zumbi
%Cpu(s): 11,0 us, 4,5 sy, 0,0 ni, 83,7 id, 0,0 wa, 0,0 hi, 0,8 si, 0,0 st
KiB Mem : 8072336 total, 5076556 free, 1615712 used, 1380068 buff/cache
KiB Swap: 8292348 total, 8292348 free, 0 used. 5803232 avail Mem
```

PID	USUÁRIO	PR	NI	VIRT	RES	SHR	S	%CPU	%MEM	TIME+	COMMAND
2767	root	20	0	453976	23760	14404	S	48,3	0,3	26:15.51	transceiver
2752	root	20	0	2134908	12564	8392	S	19,3	0,2	6:35.96	OpenBTS
3500	asterisk	20	0	2377792	23832	11160	S	4,3	0,3	0:14.38	asterisk

Fonte: A autora.

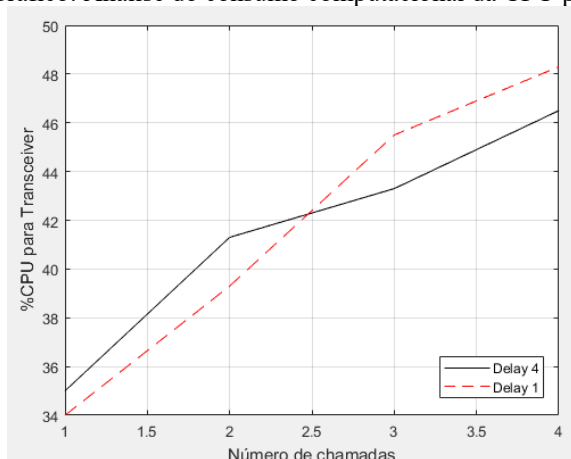
Os dados das Figuras 4.8.2, 4.8.3, 4.8.4, 4.8.5, 4.8.6, 4.8.7, 4.8.8 e 4.8.9 foram transcritos para o Quadro 4.8.1.

Quadro 4.8.1 - Custo computacional durante a realização de chamadas para a extensão 2602.

Número de Chamadas Simultâneas	Consumo Computacional (%CPU)					
	Transceiver		OpenBTS		Asterisk	
	Delay 4	Delay 1	Delay 4	Delay 1	Delay 4	Delay 1
1	35,0	34	8,7	9	0,3	0,3
2	41,3	39,3	12,7	13	1,3	1,7
3	43,3	45,5	16	16,6	2,7	2,7
4	46,5	48,3	24,9	19,3	5,3	4,3

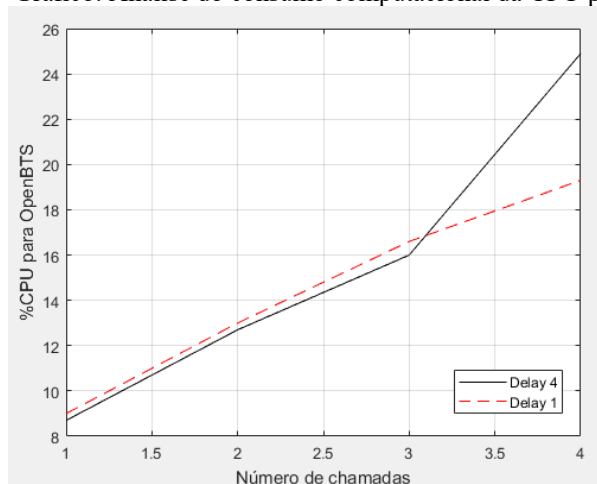
Fonte: A autora.

Os gráficos das Figuras 4.8.10, 4.8.11 e 4.8.12 foram obtidos a partir dos valores encontrados no quadro 4.8.1. O código se encontra disponível no Apêndice D.

Figura 4.8.10 - Gráfico: Análise do consumo computacional da CPU para o *Transceiver*.

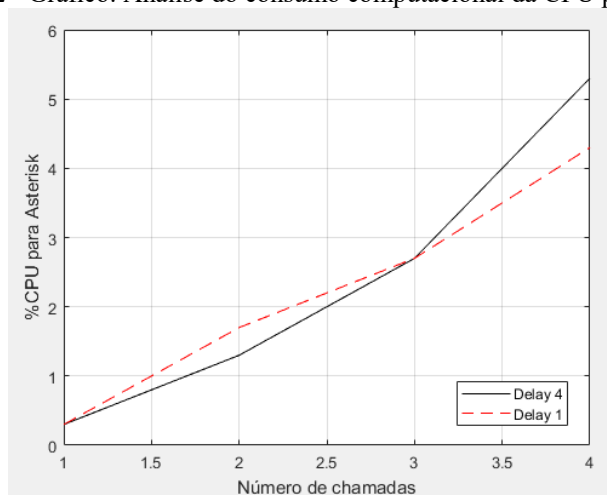
Fonte: A autora.

Figura 4.8.11 - Gráfico: Análise do consumo computacional da CPU para o *OpenBTS*.



Fonte: A autora.

Figura 4.8.12 - Gráfico: Análise do consumo computacional da CPU para o *Asterisk*.



Fonte: A autora.

Como observado nos gráficos das Figuras 4.8.10, 4.8.11 e 4.8.12, o consumo computacional da CPU (Unidade Central de Processamento) aumenta conforme o aumento do número de chamadas para ambos os processos *transceiver*, *OpenBTS* e *Asterisk*.

Observa-se no gráfico da Figura 4.8.11 que o ajuste do valor da chave *GSM.Radio.MaxExpectedDelaySpread* de 4 para 1 ocasionou uma redução do consumo computacional da CPU para o processo *OpenBTS* durante quatro chamadas simultâneas, a maior quantidade de chamadas testadas.

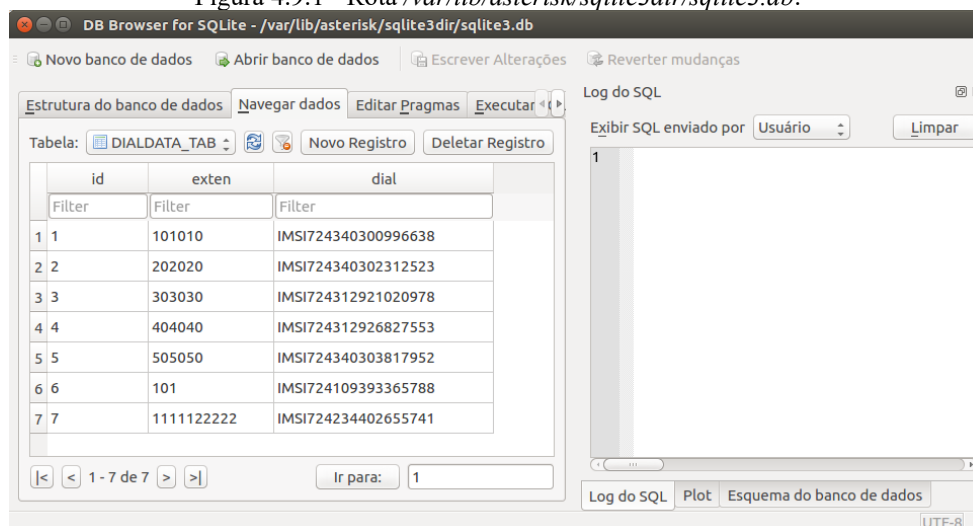
4.9 Armazenamento e Gerenciamento de Dados no *OpenBTS*

Um Sistema de Gerenciamento de Banco de Dados (SGBD) é um conjunto de *softwares* utilizado para gerenciar uma base de dados. Um exemplo de SGBD é o *SQLite* que incorpora o banco de dados SQL (*Structured Query Language*). O *SQLite* é de domínio público, ou seja, livre para uso em qualquer finalidade, seja ela comercial ou privada [26].

O *OpenBTS* utiliza o *SQLite3* para armazenar os arquivos de configurações e os dados de tempo de execução [25].

O banco de dados do registro de assinantes é utilizado pelos componentes *sipauthserve*, *smqueue* e *Asterisk*. A rota para acessá-lo é: */var/lib/asterisk/sqlite3dir/sqlite3.db*. A Figura 4.9.1 mostra uma tabela disponível nesse banco dados.

Figura 4.9.1 - Rota */var/lib/asterisk/sqlite3dir/sqlite3.db*.



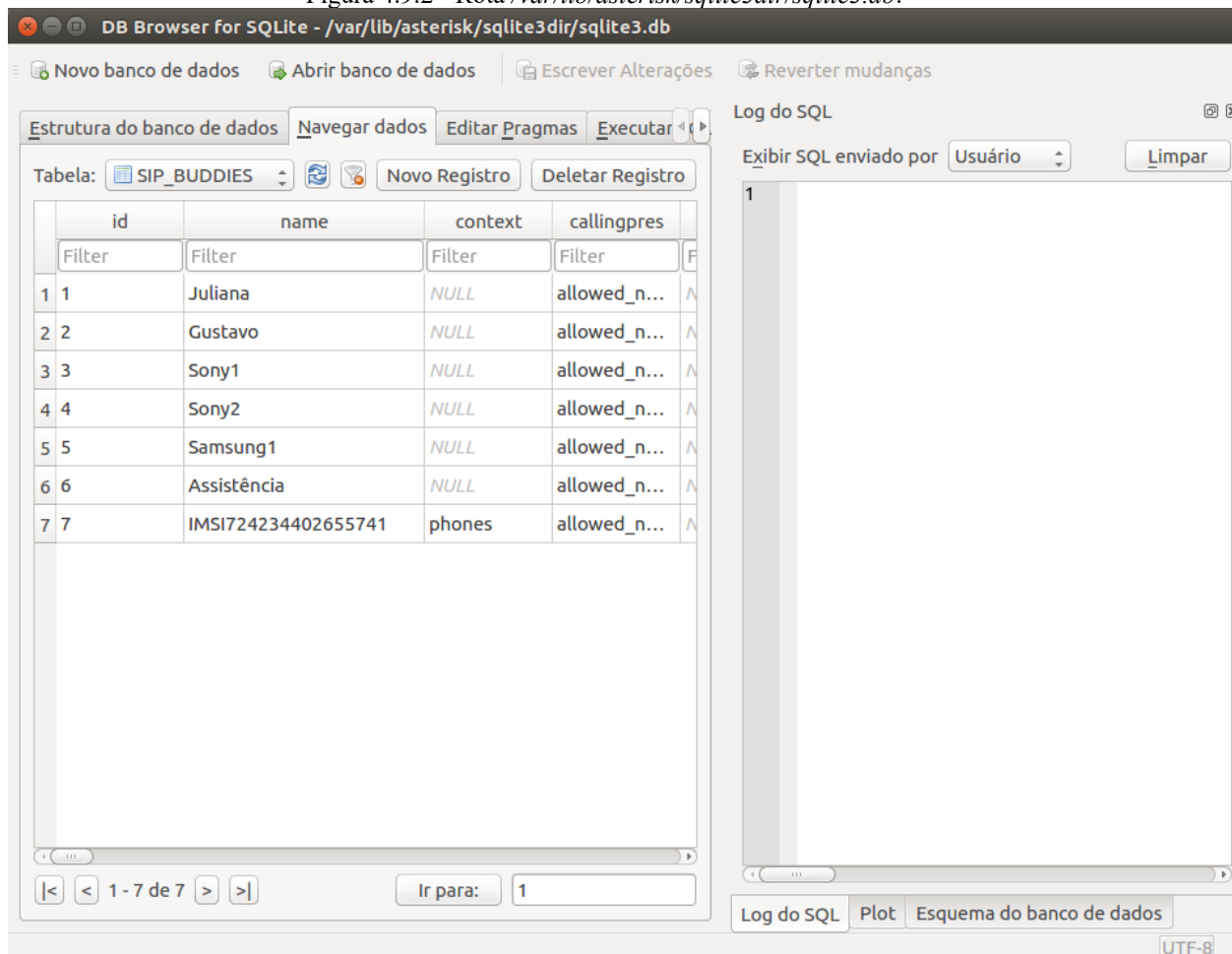
The screenshot shows the 'DB Browser for SQLite' application window. The title bar indicates the file path: */var/lib/asterisk/sqlite3dir/sqlite3.db*. The interface includes tabs for 'Estrutura do banco de dados', 'Navegar dados', 'Editar Pragas', and 'Executar'. The 'Navegar dados' tab is active, displaying a table named 'DIALDATA_TAB'. The table has three columns: 'id', 'exten', and 'dial'. Below the table, there are navigation controls and a 'Log do SQL' panel on the right.

	id	exten	dial
1	1	101010	IMSI724340300996638
2	2	202020	IMSI724340302312523
3	3	303030	IMSI724312921020978
4	4	404040	IMSI724312926827553
5	5	505050	IMSI724340303817952
6	6	101	IMSI724109393365788
7	7	111112222	IMSI724234402655741

Fonte: A autora.

Observa-se na Figura 4.9.1 uma tabela chamada “DIALDATA_TAB” contendo os dados MSISDN e IMSI de cada usuário cadastrado na rede. Verifica-se que os dados dessa tabela correspondem aos dados da Figura 4.5.6 (disponível na seção 4.5).

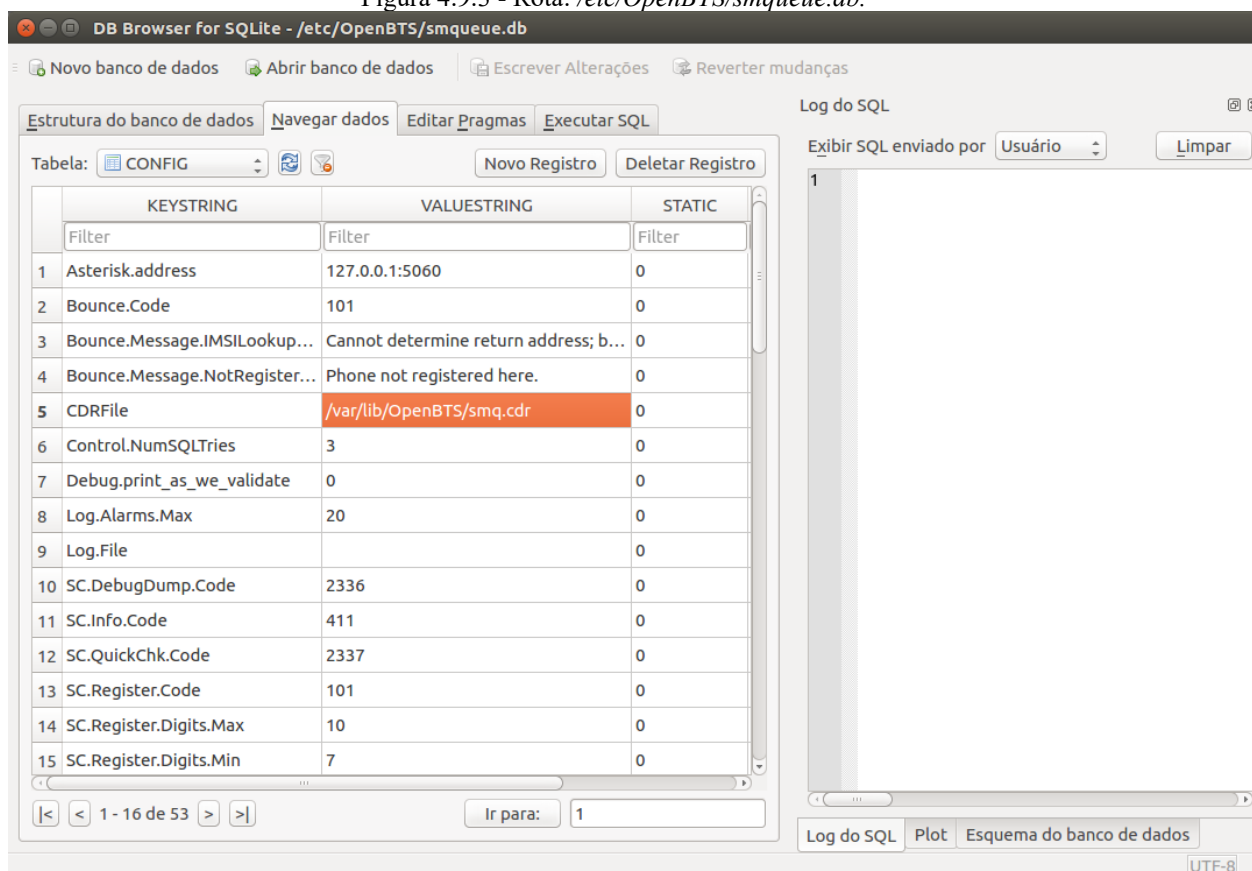
A Figura 4.9.2 mostra uma outra tabela disponível no banco de dados do registro de assinantes.

Figura 4.9.2 - Rota `/var/lib/asterisk/sqlite3dir/sqlite3.db`.

Fonte: A autora.

Observa-se na Figura 4.9.2 uma tabela chamada “SIP_BUDDIES” contendo o nome de cada usuário cadastrado na rede, verifica-se que os nomes desses usuários correspondem aos nomes dos usuários mostrados na Figura 4.5.6 (disponível na seção 4.5).

O banco de dados do *smqueue* armazena as configurações desse componente. A rota para acessá-lo é: `/etc/OpenBTS/smqueue.db` (Figura 4.9.3).

Figura 4.9.3 - Rota: */etc/OpenBTS/smqueue.db*.

Fonte: A autora.

Observa-se na Figura 4.9.3, que a rota para acessar o arquivo CDRFile é */var/lib/OpenBTS/smq.cdr*. Este arquivo mostra o registro de todos os SMSs enviados e recebidos pelos usuários da rede.

O Quadro 4.9.1 mostra um trecho do arquivo CDRFile. Observa-se que em cada linha há quatro tipos de dados separados por uma vírgula.

Quadro 4.9.1 - Registro de SMS.

MSISDN destinatário	IMSI destinatário	MSISDN remetente	Data e horário de envio do SMS
505050	IMSI724340303817952	101010	Thu Jun 21 17:29:32 2018
505050	IMSI724340303817952	101010	Thu Jun 21 17:31:06 2018
101010	IMSI724340300996638	505050	Thu Jun 21 17:31:36 2018
303030	IMSI724312921020978	505050	Thu Jun 21 17:36:12 2018
505050	IMSI724340303817952	303030	Thu Jun 21 17:38:13 2018
505050	IMSI724340303817952	303030	Thu Jun 21 17:39:38 2018
505050	IMSI724340303817952	101010	Thu Jun 21 17:39:55 2018
303030	IMSI724312921020978	505050	Thu Jun 21 17:40:09 2018
505050	IMSI724340303817952	303030	Thu Jun 21 17:40:35 2018
303030	IMSI724312921020978	505050	Thu Jun 21 17:41:01 2018
101010	IMSI724340300996638	101010	Thu Jun 21 17:42:44 2018
101010	IMSI724340300996638	505050	Thu Jun 21 17:43:05 2018
505050	IMSI724340303817952	101010	Thu Jun 21 17:43:37 2018
101010	IMSI724340300996638	505050	Thu Jun 21 17:44:13 2018
101010	IMSI724340300996638	505050	Thu Jun 21 17:44:39 2018
505050	IMSI724340303817952	101010	Thu Jun 21 17:45:07 2018
101010	IMSI724340300996638	505050	Thu Jun 21 17:45:33 2018
505050	IMSI724340303817952	101010	Thu Jun 21 17:47:04 2018
411	(null)	IMSI724340303817952	Thu Jun 21 17:49:55 2018
411	(null)	IMSI724340300996638	Thu Jun 21 17:52:34 2018
101	IMSI724109393365788	IMSI724234402655741	Mon Jun 25 13:10:32 2018
101	IMSI724109393365788	IMSI724234402655741	Mon Jun 25 13:10:50 2018
101	IMSI724109393365788	IMSI724234402655741	Mon Jun 25 13:15:22 2018

Fonte: A autora.

Analisando a primeira linha do Quadro 4.9.1, por exemplo, verifica-se que:

- 1) 505050 representa o MSISDN ao qual o SMS foi destinado;
- 2) IMSI724340303817952 representa o IMSI ao qual o SMS foi destinado;
- 3) 101010 representa o MSISDN que originou o SMS;
- 4) Thu Jun 21 17:29:32 2018 representa a data e horário que o SMS foi enviado, onde Thu representa a abreviação para *Thursday* (quinta-feira).

O registro de todas as chamadas realizadas encontra-se disponível em dois arquivos presente nas rotas: *var/log/asterisk/cdr-csv* e *var/log/asterisk/cdr-custom*.

Os Quadros 4.9.2, 4.9.3, 4.9.4 e 4.9.5 mostram um trecho do arquivo *cdr-csv*.

Os dados referentes à primeira chamada estão na primeira linha do Quadro 4.9.2 continuam na primeira linha dos Quadros 4.9.3 e 4.9.4, e terminam na primeira linha do Quadro 4.9.5.

Já os dados referentes à segunda chamada estão na primeira linha do Quadro 4.9.2, continuam na segunda linha dos Quadros 4.9.3 e 4.9.4, e terminam na segunda linha do Quadro 4.9.5. Esse padrão deve ser seguido até a última chamada.

Quadro 4.9.2 - Registro de Chamadas.

<i>src</i>	<i>dst</i>	<i>dcontext</i>	<i>clid</i>	<i>channel</i>
IMSI724312926827553	101010	to-openBTS	"404040" <IMSI724312926827553>	SIP/00101100010-0000003c
IMSI724340303817952	404040	to-openBTS	"505050" <IMSI724340303817952>	SIP/00101100010-0000003e
IMSI724312926827553	2602	default	IMSI724312926827553	SIP/00101100010-00000040
IMSI724340300996638	h-27	phones	"101010" <IMSI724340300996638>	SIP/00101100010-00000041
IMSI724340300996638	303030	to-openBTS	"101010" <IMSI724340300996638>	SIP/00101100010-00000043
IMSI724340300996638	2602	default	IMSI724340300996638	SIP/00101100010-00000045
IMSI724340300996638	2602	default	IMSI724340300996638	SIP/00101100010-00000046
IMSI724340300996638	2602	default	IMSI724340300996638	SIP/00101100010-00000047
IMSI724312921020978	2602	default	IMSI724312921020978	SIP/00101100010-00000048
IMSI724312921020978	2602	default	IMSI724312921020978	SIP/00101100010-00000049
IMSI724312921020978	2602	default	IMSI724312921020978	SIP/00101100010-0000004a
IMSI724312921020978	505050	to-openBTS	"303030" <IMSI724312921020978>	SIP/00101100010-0000004b
IMSI724312921020978	505050	to-openBTS	"303030" <IMSI724312921020978>	SIP/00101100010-0000004d
IMSI724312921020978	505050	to-openBTS	"303030" <IMSI724312921020978>	SIP/00101100010-0000004f
IMSI724340303817952	2602	default	IMSI724340303817952	SIP/00101100010-00000051
IMSI724340300996638	505050	to-openBTS	"101010" <IMSI724340300996638>	SIP/00101100010-00000052
IMSI724340300996638	h-27	phones	"101010" <IMSI724340300996638>	SIP/00101100010-00000054
IMSI724340300996638	h-27	phones	"101010" <IMSI724340300996638>	SIP/00101100010-00000056
IMSI724340300996638	h-27	phones	"101010" <IMSI724340300996638>	SIP/00101100010-00000058
IMSI724340300996638	h-27	phones	"101010" <IMSI724340300996638>	SIP/00101100010-0000005a

Fonte: A autora.

Cada um dos dados mostrados na primeira linha do Quadro 4.9.2 significam:

- 1) IMSI724312926827553: *src*
Representa o IMSI de quem originou a chamada;
- 2) 101010: *dst*
MSISDN ao qual a chamada é destinada;
- 3) to-OpenBTS: *dcontext*
Contexto de destino da chamada;
- 4) "404040" <IMSI724312926827553>: *clid*
Identificador completo de quem originou a chamada (MSISDN e IMSI);
- 5) SIP/00101100010-0000003c: *channel*
Canal utilizado por quem originou a chamada;

Quadro 4.9.3 - Registro de Chamadas (Continuação do Quadro 4.9.2).

<i>dstchannel</i>	<i>lastapp</i>	<i>lastdata</i>
SIP/127.0.0.1:5062-0000003d	Dial	SIP/IMS1724340300996638@127.0.0.1:5062,3600,g
SIP/127.0.0.1:5062-0000003f	Dial	SIP/IMS1724312926827553@127.0.0.1:5062,3600,g
	Milliwatt	
SIP/127.0.0.1:5062-00000042	Hangup	27
SIP/127.0.0.1:5062-00000044	Dial	SIP/IMS1724312921020978@127.0.0.1:5062,3600,g
	Milliwatt	
	Milliwatt	
	Milliwatt	
	Milliwatt	
	Milliwatt	
	Milliwatt	
SIP/127.0.0.1:5062-0000004c	Dial	SIP/IMS1724340303817952@127.0.0.1:5062,3600,g
SIP/127.0.0.1:5062-0000004e	Dial	SIP/IMS1724340303817952@127.0.0.1:5062,3600,g
SIP/127.0.0.1:5062-00000050	Dial	SIP/IMS1724340303817952@127.0.0.1:5062,3600,g
	Milliwatt	
SIP/127.0.0.1:5062-00000053	Dial	SIP/IMS1724340303817952@127.0.0.1:5062,3600,g
SIP/127.0.0.1:5062-00000055	Hangup	27
SIP/127.0.0.1:5062-00000057	Hangup	27
SIP/127.0.0.1:5062-00000059	Hangup	27
SIP/127.0.0.1:5062-0000005b	Hangup	27

Fonte: A autora.

Cada um dos dados mostrados na primeira linha do Quadro 4.9.3 significam:

- 1) SIP/127.0.0.1:5062-0000003d: *dstchannel*
Canal utilizado por quem recebeu a chamada;
- 2) Dial: *lastapp*
O último aplicativo *dialplan* que foi executado;
- 3) SIP/:IMS1724340300996638@127.0.0.1:5062,3600,g: *lastdata*
Os argumentos passados para o *lastapp*;

Quadro 4.9.4 - Registro de Chamadas (Continuação do Quadro 4.9.3).

<i>start</i>	<i>answer</i>	<i>end</i>	<i>duration</i>	<i>billsec</i>
2018-06-21 18:32:16	2018-06-21 18:32:25	2018-06-21 18:35:54	218	209
2018-06-21 18:38:08	2018-06-21 18:38:16	2018-06-21 18:41:12	184	176
2018-06-21 18:43:12	2018-06-21 18:43:12	2018-06-21 18:45:53	161	161
2018-06-21 18:52:10		2018-06-21 18:52:22	12	0
2018-06-21 18:53:28	2018-06-21 18:53:39	2018-06-21 18:56:57	209	198
2018-06-21 18:57:41	2018-06-21 18:57:41	2018-06-21 18:59:24	103	103
2018-06-21 18:59:29	2018-06-21 18:59:29	2018-06-21 19:00:51	82	82
2018-06-21 19:00:56	2018-06-21 19:00:56	2018-06-21 19:04:15	199	199
2018-06-21 19:05:13	2018-06-21 19:05:13	2018-06-21 19:06:31	78	78
2018-06-21 19:06:38	2018-06-21 19:06:38	2018-06-21 19:08:02	84	84
2018-06-21 19:08:18	2018-06-21 19:08:18	2018-06-21 19:10:55	157	157
2018-06-21 19:11:46	2018-06-21 19:11:53	2018-06-21 19:14:49	183	176
2018-06-21 19:16:30	2018-06-21 19:16:37	2018-06-21 19:19:32	182	175
2018-06-21 19:20:01	2018-06-21 19:20:12	2018-06-21 19:20:41	40	29
2018-06-21 19:21:59	2018-06-21 19:21:59	2018-06-21 19:25:26	207	207
2018-06-21 19:26:09	2018-06-21 19:26:17	2018-06-21 19:29:23	194	186
2018-06-21 19:30:29		2018-06-21 19:30:43	14	0
2018-06-21 19:30:53		2018-06-21 19:31:04	11	0
2018-06-21 19:31:57		2018-06-21 19:32:11	14	0
2018-06-21 19:32:22		2018-06-21 19:32:37	15	0

Fonte: A autora.

Cada um dos dados mostrados na primeira linha do Quadro 4.9.4 significam:

1) 2018-06-21 18:32:16: *start*

Data e horário de início da chamada;

2) 2018-06-21 18:32:25: *answer*

Data e horário que a chamada foi atendida;

3) 2018-06-21 18:35:54: *end*

Data e horário que a chamada foi finalizada;

4) 218: *duration*

Quantidade de segundos entre o início e o fim da chamada;

5) 209: *billsec*

Quantidade de segundos entre o atendimento e o fim da chamada.

Quadro 4.9.5 - Registro de Chamadas (Continuação do Quadro 4.9.4).

<i>disposition</i>	<i>amaflags</i>	<i>uniqueid</i>
ANSWERED	DOCUMENTATION	1529605936.60
ANSWERED	DOCUMENTATION	1529606288.62
ANSWERED	DOCUMENTATION	1529606592.64
FAILED	DOCUMENTATION	1529607130.65
ANSWERED	DOCUMENTATION	1529607208.67
ANSWERED	DOCUMENTATION	1529607461.69
ANSWERED	DOCUMENTATION	1529607569.70
ANSWERED	DOCUMENTATION	1529607656.71
ANSWERED	DOCUMENTATION	1529607913.72
ANSWERED	DOCUMENTATION	1529607998.73
ANSWERED	DOCUMENTATION	1529608098.74
ANSWERED	DOCUMENTATION	1529608306.75
ANSWERED	DOCUMENTATION	1529608590.77
ANSWERED	DOCUMENTATION	1529608801.79
ANSWERED	DOCUMENTATION	1529608919.81
ANSWERED	DOCUMENTATION	1529609169.82
FAILED	DOCUMENTATION	1529609429.84
FAILED	DOCUMENTATION	1529609453.86
FAILED	DOCUMENTATION	1529609517.88
FAILED	DOCUMENTATION	1529609542.90

Fonte: A autora.

Cada um dos dados mostrados na primeira linha do Quadro 4.9.5 significam:

1) ANSWERED: *disposition*

Indicação do que aconteceu a chamada, pode ser NO ANSWER, FAILED, BUSY, ANSWERED ou UNKNOWN;

2) DOCUMENTATION: *amaflags*

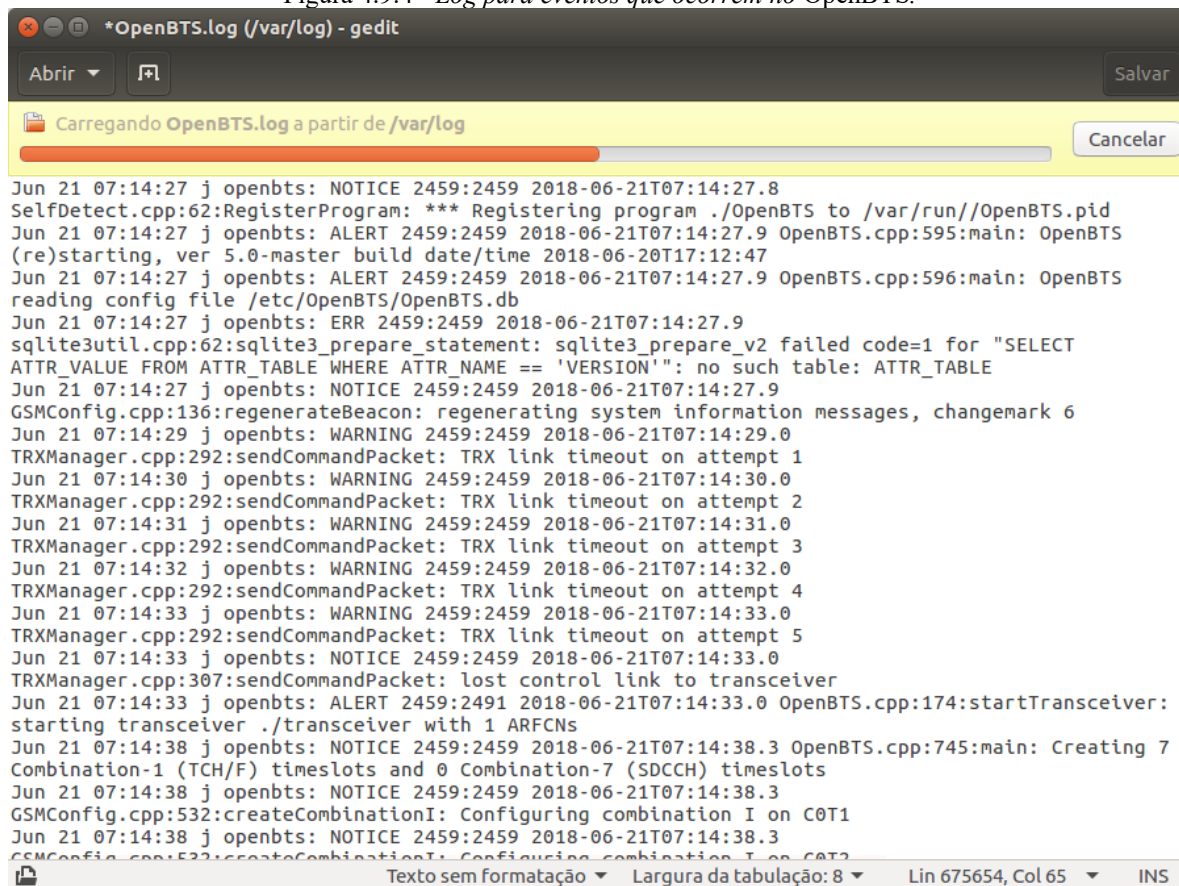
Sinalizador AMA (*Automatic Message Accounting*) associado a esta chamada. Fornece contabilidade detalhada para as chamadas telefônicas. Pode ser: *OMIT*, *BILLING*, *DOCUMENTATION* ou *Unknown*;

3) 159605936.60: *uniqueid*

ID exclusivo do canal *src* (canal de quem originou a chamada).

O destino do *log* para os eventos de todos os componentes do *OpenBTS* encontra-se na rota: */var/log/OpenBTS.log*. A Figura 4.9.4 mostra um pequeno trecho de *log* armazenado.

Figura 4.9.4 - Log para eventos que ocorrem no OpenBTS.



Fonte: A autora.

Para acessar o *log* gerado em tempo real para os componentes do *OpenBTS*, deve executar-se em um novo terminal:

```
$ tail -f /var/log/OpenBTS.log
```

Para monitorar entradas de *log* apenas para o componente *sipauthserve*, por exemplo, executa-se o comando:

```
$ tail -f /var/log/OpenBTS.log | grep sipauthserve
```

As Figura 4.9.5 mostra uma entrada de *log* gerada para o componente *sipauthserve*, quando o aparelho celular com cartão SIM de IMSI 724234402655741 conecta-se na rede.

Figura 4.9.5 - Entrada de *log* quando aparelho celular conecta-se à rede.

```
openbts@j:~$ tail -f /var/log/OpenBTS.log | grep sipauthserve
Jun 28 12:29:42 j sipauthserve: NOTICE 22629:22629 2018-06-28T12:29:42.2 sipauth
serve.cpp:234:processBuffer: imsi 724234402655741 known, 1st register
```

Fonte: A autora.

O comando a seguir deve ser executado para monitorar entradas de *log* para o componente *smqueue*.

```
$ tail -f /var/log/OpenBTS.log | grep smqueue
```

A Figura 4.9.6 mostra todas as entradas de *log* geradas para o componente *smqueue*, durante a troca de duas mensagens SMSs entre dois aparelhos celulares.

Figura 4.9.6 - Entradas de *log* geradas durante troca de SMSs.

```
openbts@j:~$ tail -f /var/log/OpenBTS.log | grep smqueue
Jun 28 13:19:43 j smqueue: NOTICE 3197:3203 2018-06-28T13:19:43.9 smqueue.cpp:2455:main
_loop: Got SMS rqst qtag '758464--OBTSzjxtclpxygvzdqoa' from IMSI724340303817952 for sm
sc
Jun 28 13:19:43 j smqueue: NOTICE 3197:3204 2018-06-28T13:19:43.9 smqueue.h:505:get_tex
t: Decoded text: Oi! Tudo bem?
Jun 28 13:19:44 j smqueue: NOTICE 3197:3204 2018-06-28T13:19:44.9 smqueue.h:505:get_tex
t: Decoded text: Oi! Tudo bem?
Jun 28 13:21:04 j smqueue: NOTICE 3197:3204 2018-06-28T13:21:04.6 smqueue.cpp:318:handl
e_response: Got 200 response for sent msg '758464--OBTSzjxtclpxygvzdqoa' in state 12
Jun 28 13:22:27 j smqueue: NOTICE 3197:3203 2018-06-28T13:22:27.2 smqueue.cpp:2455:main
_loop: Got SMS rqst qtag '758467--OBTSrpmilipaninmsc' from IMSI724340300996638 for sm
sc
Jun 28 13:22:27 j smqueue: NOTICE 3197:3204 2018-06-28T13:22:27.2 smqueue.h:505:get_tex
t: Decoded text: Ola! Estou bem!
Jun 28 13:22:28 j smqueue: NOTICE 3197:3204 2018-06-28T13:22:28.2 smqueue.h:505:get_tex
t: Decoded text: Ola! Estou bem!
Jun 28 13:22:30 j smqueue: NOTICE 3197:3204 2018-06-28T13:22:30.8 smqueue.cpp:318:handl
e_response: Got 200 response for sent msg '758467--OBTSrpmilipaninmsc' in state 12
```

Fonte: A autora.

Para pesquisar em um arquivo de *log* uma palavra específica, como por exemplo, pesquisar em todo o arquivo de *log* a palavra *sipauthserve*, executa-se o comando:

```
$ grep sipauthserve /var/log/OpenBTS.log
```

5 CONCLUSÃO

O desenvolvimento desse trabalho proporcionou um estudo mais abrangente do Sistema de Telefonia Celular, da tecnologia GSM (2G) e do *Asterisk*. Além disso, permitiu o contato com a tecnologia de Rádio Definido por *Software*, com o Projeto *OpenBTS* e com a distribuição *Linux Ubuntu*. Todos os *softwares* utilizados são *open source*.

É perceptível que são muitos os benefícios em utilizar as tecnologias voltadas para *Software*. Essas tecnologias são economicamente viáveis devido ao fato da quantidade de equipamentos de *hardware* ser reduzida, e além disso a manutenção dos sistemas implementados ser feita com atualizações de *software*, o que a torna muito mais flexível. Uma rede móvel GSM implementada em *software* usufrui de todos esses benefícios.

Com a Estação Rádio Base implementada através do *OpenBTS*, os serviços de chamada de voz e mensagens SMS foram realizados com sucesso. Toda a configuração realizada foi mostrada detalhadamente. Além disso, foi realizada uma descrição sucinta de como implementar várias ERBs, e assim expandir a área de cobertura.

Foram realizados testes para medir a qualidade do *link* entre as estações móveis e a ERB. Verificou-se as frequências da portadora em um equipamento Analisador de Espectro. Realizou-se uma análise do custo computacional durante as chamadas. Descreveu-se onde são armazenados os dados de tempo de execução e os arquivos de configurações no banco de dados utilizado pelo *OpenBTS*. Mostrou-se detalhadamente os registros de chamadas e de SMS.

Esse trabalho proporcionou o aprendizado de inúmeros conceitos da área de Telecomunicações. Sendo possível ver na prática muitos conteúdos abordados no decorrer de algumas das disciplinas do curso de Engenharia Eletrônica e de Telecomunicações.

Trabalhos Futuros

O trabalho desenvolvido atingiu o objetivo proposto. Entretanto, uma melhoria a ser realizada é a tarifação das chamadas utilizando o *software A2Billing* [28].

Para continuar trabalhando com Estações Rádio Base utilizando Rádio Definido por *Software* e as tecnologias *open source*, implementar uma ERB utilizando o projeto *OpenBTS-UMTS* (3G) [29] e/ou uma ERB utilizando o projeto *OpenLTE* (4G) [30].

REFERÊNCIAS

- [1] RAPPAPORT, Theodore S. **Comunicações sem fio: princípios e práticas**. 2 ed. São Paulo: Pearson Prentice Hall, 2009.
- [2] IEDEMA, Michael. **Getting Started with OpenBTS: Build Open Source Mobile Networks**. " O'Reilly Media, Inc.", 2015.
- [3] ROUPHAEL, Tony J. **RF and digital signal processing for software-defined radio: a multi-standard multi-mode approach**. Newnes, 2009.
- [4] WANDER. Unidade 11: **Comunicações Móveis** (Apostila). CEFET-MG, 2003. Disponível em <<http://www.jdbte.com.br/wjrteleco/unit%2011.pdf>>. Acesso em: 26/05/2017.
- [5] STASIAK, Maciej et al. **Modelling and dimensioning of mobile wireless networks: from GSM to LTE**. John Wiley & Sons, 2010.
- [6] MISHRA, Ajay R. (Ed.). **Advanced cellular network planning and optimisation: 2G/2.5 G/3G... evolution to 4G**. John Wiley & Sons, 2007.
- [7] What is Software Defined Radio? Disponível em: <http://www.wirelessinnovation.org/index.php?option=com_content&view=article&id=63:Introduction_to_SDR&catid=19:site-content&Itemid=77>. Acesso em: 26/05/2017.
- [8] ROCHA, Gustavo Nozella et al. Separação cega de fontes aplicada no sensoriamento do espectro em rádio cognitivo. 2012.
- [9] What is Asterisk? Disponível em: <<http://www.asterisk.org/get-started>>. Acesso em: 26/05/2017.
- [10] What is a PBX Phone System? Disponível em: <<https://www.3cx.com.br/voip-sip/sistema-telefonias-pbx/>>. Acesso em: 26/05/2017.
- [11] MADSEN, Leif; VAN MEGGELEN, Jim; BRYANT, Russell. **Asterisk: The definitive guide**. " O'Reilly Media, Inc.", 2013.
- [12] SANKHE, Kunal et al. **Cost effective restoration of wireless connectivity in disaster hit areas using OpenBTS**. In: India Conference (INDICON), 2014 Annual IEEE. IEEE, 2014. p. 1-6.
- [13] CHENG, Shin-Ming et al. **Experimental emergency communication systems using USRP and GNU radio platform**. In: Heterogeneous Networking for Quality, Reliability, Security and Robustness (QSHINE), 2015 11th International Conference on. IEEE, 2015. p. 75-79.

- [14] CABRAL, Marcel et al. **Low-cost gsm telephony in the amazon region based on open-source/open-hardware projects**. In: Communications, 2009. LATINCOM'09. IEEE Latin-American Conference on. IEEE, 2009. p. 1-6.
- [15] HATORANGAN, Elvanno; JUHANA, Tutun. **Mobile phone location logging into OpenBTS-based cellular network in disaster situation**. In: Telecommunication Systems Services and Applications (TSSA), 2014 8th International Conference on. IEEE, 2014. p. 1-3.
- [16] File:Product n210.jpg. Disponível em: < https://kb.ettus.com/File:Product_n210.jpg>. Acesso em: 26/05/2017.
- [17] N200/N210. Disponível em: <<https://kb.ettus.com/N200/N210>>. Acesso em: 26/05/2017.
- [18] USRP N200/N210 NETWORKED SERIES. Disponível em: <https://www.ettus.com/content/files/07495_Ettus_N200-210_DS_Flyer_HR_1.pdf>. Acesso em: 26/05/2017.
- [19] Seção: Tutoriais Telefonia Celular. Disponível em: <http://www.teleco.com.br/tutoriais/tutorialbandcel/pagina_5.asp>. Acesso em: 24/06/2018.
- [20] Introdução ao JSON. Disponível em: <<https://www.json.org/json-pt.html>>. Acesso em: 24/06/2018.
- [21] Wireless Security. Disponível em: <<http://www.ic.unicamp.br/~edmundo/MO818/turma2s2005/semfio/WirelessSecurity.pdf>>. Acesso em: 24/06/2018.
- [22] GSM. Disponível em: <http://www.telecom.uff.br/pagina/posgraduacao/Lato-Sensu/uploads/6/9/4/8/6948141/comunicaes_mveis_-_parte_2_-_gsm_rev_2012.pdf>. Acesso em: 24/06/2018.
- [23] RejectCauses. Disponível em: <<http://openbts.org/w/index.php?title=RejectCauses>>. Acesso em: 25/06/2018.
- [24] Mobile Country Codes (MCC) and Mobile Network Codes (MNC). Disponível em: <<http://www.mcc-mnc.com/>>. Acesso em: 25/06/2018.
- [25] OpenBTS Application Suite User Manual. Version 4.0. Release, 2014. Disponível em: <<http://openbts.org/site/wp-content/uploads/2014/07/OpenBTS-4.0-Manual.pdf>>.
- [26] TOP 10 principais SGBDs do mercado global! Disponível em: <<https://becode.com.br/principais-sgbds/>>. Acesso em: 26/06/2018.
- [27] Call Detail Records. Disponível em: <http://www.asteriskdocs.org/en/3rd_Edition/asterisk-book-html-chunk/asterisk-SysAdmin-SECT-1.html>. Acesso em: 28/06/2018.
- [28] A2Billing. Disponível em: <<http://www.asterisk2billing.org/>>. Acesso em: 28/06/2018.

[29] OpenBTS-UMTS. Disponível em: <<http://openbts.org/w/index.php?title=OpenBTS-UMTS>>. Acesso em: 28/06/2018.

[30] OpenLTE. Disponível em: < <http://openlte.sourceforge.net/> >. Acesso em: 28/06/2018.

APÊNDICE A – Instalação do *OpenBTS* e seus componentes

Todos os comandos para a instalação do *OpenBTS* e seus componentes são baseados em [2], porém, com algumas modificações, visto que alguns comandos dessa fonte se encontram obsoletos. Além disso, alguns desses comandos são para instalação em sistema operacional de 32 bits e nesse trabalho é utilizado um sistema operacional de 64 bits.

O projeto *OpenBTS* utiliza vários recursos do *Git*, um sistema de controle de versão que gerencia mudanças no código fonte de diversos *softwares*. Execute o comando a seguir, para adicionar suporte aos Arquivos de Pacotes Pessoais (*Personal Package Archives* - PPA).

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

Para adicionar um repositório para as compilações *Git* mais recentes execute o comando:

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

Para atualizar a lista de pacotes e instalar o *Git* execute os seguintes comandos:

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

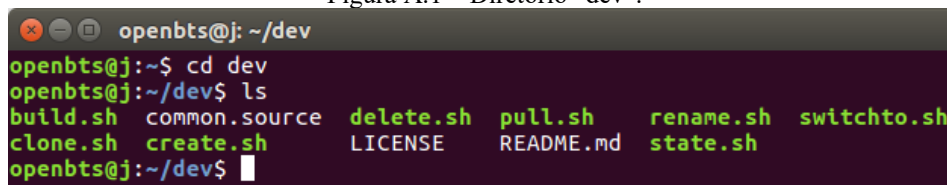
O projeto *OpenBTS* consiste em vários componentes de *software* que estão em repositórios separados no *GitHub*. Alguns *scripts* de desenvolvimento foram escritos para facilitar o *download* desses componentes. Para fazer o *download* desses *scripts* execute o comando:

```
$ git clone https://github.com/RangeNetworks/dev.git
```

Após isso, é criado um diretório chamado “*dev*”, como mostra a Figura A.1. Para entrar nesse diretório e listar seu conteúdo, execute os comandos:

```
$ cd dev  
$ ls
```

Figura A.1 – Diretório “dev”.



```

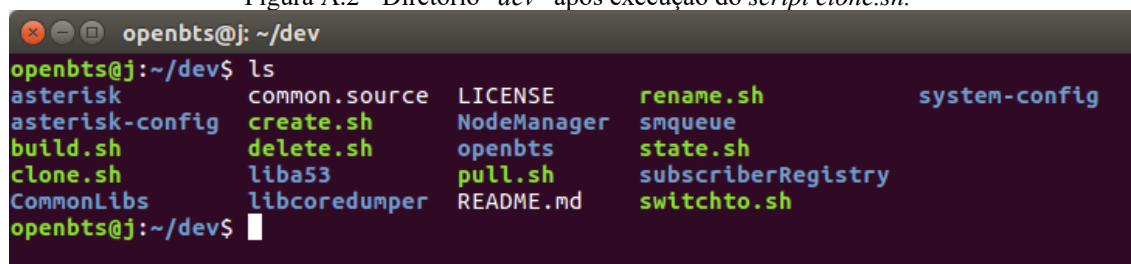
openbts@j: ~/dev
openbts@j:~$ cd dev
openbts@j:~/dev$ ls
build.sh  common.source  delete.sh  pull.sh  rename.sh  switchto.sh
clone.sh  create.sh      LICENSE    README.md  state.sh
openbts@j:~/dev$

```

Fonte: A autora.

Dentro do diretório “dev” execute o *script clone.sh* para fazer *download* de todos os componentes que integram o *OpenBTS*. A Figura A.2 mostra o diretório “dev” após a execução desse *script*.

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

Figura A.2 - Diretório “dev” após execução do *script clone.sh*.


```

openbts@j: ~/dev
openbts@j:~/dev$ ls
asterisk      common.source  LICENSE      rename.sh    system-config
asterisk-config  create.sh      NodeManager  smqueue
build.sh      delete.sh      openbts      state.sh
clone.sh      liba53         pull.sh      subscriberRegistry
CommonLibs    libcoredumper  README.md    switchto.sh
openbts@j:~/dev$

```

Fonte: A autora.

Após as fontes do projeto do *OpenBTS* estarem no ambiente de desenvolvimento, é preciso escolher uma versão do *OpenBTS* para compilar. Dentro do diretório “dev”, execute o comando a seguir para compilar a versão *master* do *OpenBTS*.

```
$ ./switchto.sh master
```

Para compilar pacotes binários deve-se usar o *script build.sh*. Esse *script* instala automaticamente as ferramentas de configuração e autoconfiguração, todas as dependências necessárias e controla qual aplicativo transceptor de rádio será construído.

Existem vários *drivers* diferentes disponíveis para os vários tipos de rádio. Sendo assim, o *build.sh* requer um argumento para saber ao qual *hardware* está sendo direcionado. Como nesse trabalho foi utilizado o *hardware* USRP N210, ainda dentro do diretório “dev”, execute o seguinte comando:

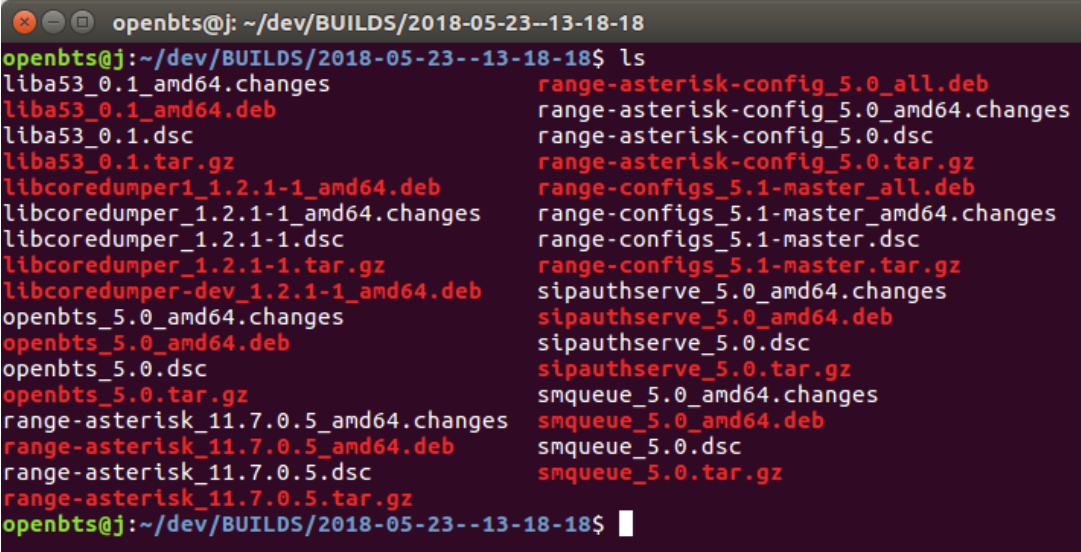
```
$ ./build.sh N210
```

Um novo diretório chamado “*BUILDS*” contendo um subdiretório com o *timestamp* (data e horário) da compilação é criado após o término da execução do comando *./build.sh N210*. Para entrar nesse diretório e listar seu conteúdo execute os seguintes comandos:

```
$ cd BUILDS/2018-05-23--13-18-18/
$ ls
```

A Figura A.3 mostra o diretório *BUILDS/2018-05-23--13-18-18/*. É nesse diretório que os comandos para as instalações dos componentes devem ser executados.

Figura A.3 – Diretório *BUILDS/2018-05-23--13-18-18/*.



```
openbts@j:~/dev/BUILDS/2018-05-23--13-18-18$ ls
liba53_0.1_amd64.changes      range-asterisk-config_5.0_all.deb
liba53_0.1_amd64.deb         range-asterisk-config_5.0_amd64.changes
liba53_0.1.dsc               range-asterisk-config_5.0.dsc
liba53_0.1.tar.gz            range-asterisk-config_5.0.tar.gz
libcoredumper1_1.2.1-1_amd64.deb range-configs_5.1-master_all.deb
libcoredumper1_1.2.1-1_amd64.changes range-configs_5.1-master_amd64.changes
libcoredumper1_1.2.1-1.dsc   range-configs_5.1-master.dsc
libcoredumper1_1.2.1-1.tar.gz range-configs_5.1-master.tar.gz
libcoredumper-dev_1.2.1-1_amd64.deb sipauthserve_5.0_amd64.changes
openbts_5.0_amd64.changes    sipauthserve_5.0_amd64.deb
openbts_5.0_amd64.deb        sipauthserve_5.0.dsc
openbts_5.0.dsc              sipauthserve_5.0.tar.gz
openbts_5.0.tar.gz           smqueue_5.0_amd64.changes
range-asterisk_11.7.0.5_amd64.changes smqueue_5.0_amd64.deb
range-asterisk_11.7.0.5_amd64.deb  smqueue_5.0.dsc
range-asterisk_11.7.0.5.dsc       smqueue_5.0.tar.gz
range-asterisk_11.7.0.5.tar.gz
openbts@j:~/dev/BUILDS/2018-05-23--13-18-18$
```

Fonte: A autora.

É necessária a instalação de duas bibliotecas, a *Coredumper* e a *A5/3*. A biblioteca *Coredumper* é utilizada para produzir informações de depuração se o *OpenBTS* travar. Já a biblioteca *A5/3* é utilizada para suportar a criptografia de chamadas. Execute os comandos a seguir para instalar ambas as bibliotecas:

```
$ sudo dpkg -i libcoredumper1_1.2.1-1_amd64.deb
$ sudo apt-get -f install
$ sudo dpkg -i liba53_0.1_amd64.deb
$ sudo apt-get -f install
```

A instalação dos demais componentes, permite ter uma rede GSM totalmente funcional com os serviços de voz, SMS e dados.

Para instalar o pacote de configurações do sistema para a interface de rede, regras de *firewall*, configuração do Sistema de Nomes de Domínio (*Domain Name System* - DNS), criação de *log*, etc., execute os comandos a seguir respondendo “Y” a cada solicitação de confirmação de sobrescrita de determinados arquivos de configuração.

```
$ sudo dpkg -i range-configs_5.1-master_all.deb
$ sudo apt-get -f install
```

Os pacotes *range-asterisk* e *range-asterisk-configs* são responsáveis por configurar a instalação do *Asterisk* para funcionar sem qualquer configuração adicional.

O pacote *range-asterisk* contém uma versão de funcionamento confirmada do *software* do *switch Asterisk SIP* e garante que os módulos apropriados necessários para o *OpenBTS* já estejam incluídos [2].

Já o pacote *range-asterisk-configs* contém um conjunto de arquivos de configuração para que o *Asterisk* conheça e possa se comunicar com o banco de dados do registro de assinantes. É nesse banco de dados que os componentes armazenam e atualizam os números de telefone, identidades, autenticações, identificadores de chamadas e estados de registro dos assinantes [2].

Ao usar o banco de dados do registro de assinantes, não é mais necessário editar manualmente os arquivos de configuração do *Asterisk* ao adicionar novos aparelhos à rede.

Para as instalações dos pacotes relacionados ao *Asterisk* execute os comandos:

```
$ sudo dpkg -i range-asterisk*.deb
$ sudo apt-get install -f
```

Para instalar o *SIPAuthServe* execute os seguintes comandos:

```
$ sudo dpkg -i sipauthserve_5.0_amd64.deb
$ sudo apt-get install -f
```

Para instalar o *SMQueue* execute os seguintes comandos:

```
$ sudo dpkg -i smqueue_5.0_amd64.deb
$ sudo apt-get install -f
```

Para a instalar o *OpenBTS* execute os seguintes comandos:

```
sudo dpkg -i openbts_5.0_amd64.deb
sudo apt-get install -f
```

Após a execução de todos os comandos anteriores é possível acessar a raiz do *OpenBTS* (Figura A.4), local onde se encontram os principais componentes, com exceção do *Asterisk*, que são utilizados para o funcionamento da rede móvel GSM. Para acessar a raiz do *OpenBTS* execute:

```
$ cd /OpenBTS/
$ ls
```

Figura A.4 - Raiz do *OpenBTS*.

```

openbts@j: /OpenBTS
openbts@j:~$ cd /OpenBTS/
openbts@j:/OpenBTS$ ls
comp128  OpenBTS  OpenBTSCLI  OpenBTSDo  sipauthserve  smqueue  transceiver
openbts@j:/OpenBTS$

```

Fonte: A autora.

O *OpenBTS* e todos os seus componentes também se encontram no diretório “dev” e seus subdiretórios. Para acessar o *OpenBTS* pelo diretório “dev”, como mostra a Figura A.5, execute o seguinte comando:

```
$ cd /dev/openbts/apps/
```

Figura A.5 - Acesso ao *OpenBTS* pelo diretório “dev”.

```

openbts@j: ~/dev/openbts/apps
openbts@j:~$ cd dev
openbts@j:~/dev$ ls
asterisk          common.source  NodeManager    state.sh
asterisk-config  create.sh      openbts        subscriberRegistry
BUILDS           delete.sh     pull.sh        switchto.sh
build.sh         liba53        README.md      system-config
clone.sh         libcoredumper rename.sh
CommonLibs       LICENSE       smqueue
openbts@j:~/dev$ cd openbts/
openbts@j:~/dev/openbts$ ls
aclocal.m4        config.log      GSM             NodeManager
apps              config.status  GSMShare       package
AUTHORS           config.sub     INSTALL        Peering
autogen.sh        configure      INSTALLATION   README
build-arch-stamp  configure.ac   install-sh     README.APIs.md
build-indep-stamp configure-stamp LEGAL           Scanning
ChangeLog         Control        libtool        SGSNGGSN
CLI               COPYING        ltmain.sh      SIP
CommonLibs        ctags.sh      Makefile        SMS
compile           debian         Makefile.am     stamp-h1
config            depcomp        Makefile.common tools
config.guess      doc            Makefile.in     Transceiver52M
config.h          Globals        missing          TransceiverRAD1
config.h.in       GPRS          NEWS            TRXManager
openbts@j:~/dev/openbts$ cd apps
openbts@j:~/dev/openbts/apps$ ls
CLI               JSONEventsClient.cpp  OpenBTSCLI-OpenBTSCLI.o
exportConfigTable.sh  JSONEventsClient.o   openbts.conf
generateConfigTable.sh logrotated.OpenBTS   openbtsconfig
generateTeX.sh        Makefile              OpenBTSCfg.h
GetConfigurationKeys.cpp  Makefile.am          OpenBTS.cpp
GetConfigurationKeys.o  Makefile.in           OpenBTSDo
importConfigTable.sh   OpenBTS               OpenBTS.example.sql
iptables.rules         OpenBTSCLI            OpenBTS.o
JSONEventsClient       OpenBTSCLI.cpp        rsyslogd.OpenBTS.conf
openbts@j:~/dev/openbts/apps$

```

Fonte: A autora.

Exceto o USRP1, os demais dispositivos USRP, utilizam o *Universal Hardware Driver* (UHD) para suporte unificado de *firmware* e *software host* [31]. O *driver* UHD já se encontra integrado aos componentes do *OpenBTS*, porém, é necessário fazer o *download* das imagens UHD. Os comandos relacionados ao UHD são originários de [32].

Utilize os seguintes comandos para fazer o *download* das imagens UHD:

```
$ sudo su
# cd /usr/lib/uhd/utils/
# ls
# uhd_images_downloader
```

O local de destino dessas imagens estará na seguinte rota:

```
# cd /usr/share/uhd/images
```

A Figura A.6 mostra a execução dos comandos para o *driver* UHD.

Figura A.6 - Execução dos comandos para o *driver* UHD.

```
openbts@j:~$ sudo su
[sudo] senha para openbts:
root@j:/home/openbts# cd /usr/lib/uhd/utils/
root@j:/usr/lib/uhd/utils# ls
b2xx_fx3_utils      uhd-usrp.rules      usrp_n2xx_net_burner_gui.py
fx2_init_eeeprom    usrp2_card_burner_gui.py  usrp_n2xx_net_burner.py
latency             usrp2_card_burner.py     usrp_n2xx_simple_net_burner
octoclock_burn_eeeprom  usrp2_recovery.py       usrp_x3xx_fpga_burner
query_gpsdo_sensors  usrp_burn_db_eeeprom
uhd_images_downloader.py  usrp_burn_nb_eeeprom
root@j:/usr/lib/uhd/utils# uhd_images_downloader
Images destination: /usr/share/uhd/images
Downloading images from: http://files.ettus.com/binaries/images/uhd-images_003.009.002-release.zip
Downloading images to: /tmp/tmpozxRdf/uhd-images_003.009.002-release.zip
26296 kB / 26296 kB (100%)

Images successfully installed to: /usr/share/uhd/images
root@j:/usr/lib/uhd/utils# cd
root@j:~# cd /usr/share/uhd/images/
root@j:/usr/share/uhd/images# ls
003.009.002.tag      usrp2_fpga.bin      usrp_b205mini_fpga.bin  usrp_e3xx_fpga_idle_sg3.bit  usrp_n210_r4_fpga.bin
bit                 usrp2_fw.bin        usrp_b210_fpga.bin      usrp_n200_fw.bin            usrp_x300_fpga_HGS.bit
LICENSE             usrp_b100_fpga_2rx.bin  usrp_e100_fpga_v2.bin   usrp_n200_r2_fpga.bin       usrp_x300_fpga_HGS.lvbitx
octoclock_bootloader.hex  usrp_b100_fpga.bin    usrp_e110_fpga.bin      usrp_n200_r3_fpga.bin       usrp_x310_fpga_HGS.bit
octoclock_r4_fw.hex      usrp_b100_fw.ihx      usrp_e310_fpga.bit      usrp_n200_r4_fpga.bin       usrp_x310_fpga_HGS.lvbitx
usrp1_fpga_4rx.rbf       usrp_b200_fpga.bin    usrp_e310_fpga_idle.bit  usrp_n210_fw.bin            winusb_driver
usrp1_fpga.rbf          usrp_b200_fw.hex      usrp_e310_fpga_sg3.bit  usrp_n210_r2_fpga.bin
usrp1_fw.ihx           usrp_b200mini_fpga.bin  usrp_e3xx_fpga_idle.bit  usrp_n210_r3_fpga.bin
root@j:/usr/share/uhd/images#
```

Fonte: A autora.

É possível observar na Figura A.6 que foi feito o *download* das imagens UHD versão 003.009.002. É esta versão que é compatível com a versão do *OpenBTS* instalada.

Referências

[31] Ettus Research USRP. Disponível em:
< http://openbts.org/w/index.php?title=Ettus_Research_USRP>. Acesso em: 02/06/2018.

[32] Firmware and FPGA Images. Disponível em:
<https://files.ettus.com/manual/page_images.html>. Acesso em: 02/06/2018.

APÊNDICE B – Conexão do *host* e do aplicativo *transceiver* com o USRP N210

Após fazer o *download* das imagens UHD é necessário configurar a conexão com o USRP N210. Os comandos e configurações aqui descritos são baseados em [33].

O endereço IPv4 padrão do USRP N210 é 192.168.10.2. O Quadro B.1 apresenta os parâmetros para configuração da conexão *Gigabit Ethernet*. A Figura B.1 mostra a configuração desta conexão e a Figura B.2 mostra a conexão de rede ativa do USRP N210 com o *host*.

Quadro B.1 - Parâmetros de configuração da conexão *Gigabit Ethernet*.

Endereço IPv4 estático para o <i>host</i>	192.168.10.1
Máscara de sub-rede	255.255.255.0
<i>Gateway</i>	0.0.0.0

Fonte: [22] (Adaptado).

Figura B.1 - Configuração da conexão *Gigabit Ethernet*.

Nome da conexão: USRP N210

Geral Ethernet Segurança 802.1x DCB Configurações IPv4 Configurações IPv6

Método: Manual

Endereços

Endereço	Máscara de rede	Gateway
192.168.10.1	255.255.255.0	0.0.0.0

Adicionar Excluir

Servidores DNS:

Pesquisar domínios:

ID do cliente DHCP:

☒ Requer endereçamento de IPv4 para que esta conexão seja concluída

Rotas...

Cancelar Salvar

Fonte: A autora.

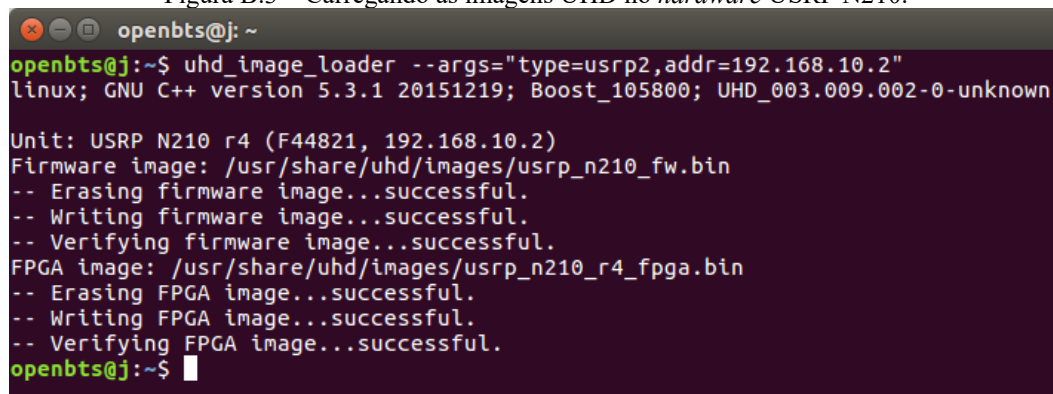
Figura B.2 - Conexão de rede ativa do USRP N210 com o *host*.

Fonte: A autora.

Após conectar o USRP N210 com o *host* é necessário carregar as imagens UHD do *firmware* e da FPGA no *hardware* USRP N210 como mostra a Figura B.3. Isso é feito executando o seguinte comando:

```
$ uhd_image_loader --args="type=usrp2,addr=192.168.10.2"
```

Depois de executar o comando anterior, o dispositivo USRP N210 deve ser reinicializado.

Figura B.3 – Carregando as imagens UHD no *hardware* USRP N210.


```

openbts@j: ~
openbts@j:~$ uhd_image_loader --args="type=usrp2,addr=192.168.10.2"
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_003.009.002-0-unknown

Unit: USRP N210 r4 (F44821, 192.168.10.2)
Firmware image: /usr/share/uhd/images/usrp_n210_fw.bin
-- Erasing firmware image...successful.
-- Writing firmware image...successful.
-- Verifying firmware image...successful.
FPGA image: /usr/share/uhd/images/usrp_n210_r4_fpga.bin
-- Erasing FPGA image...successful.
-- Writing FPGA image...successful.
-- Verifying FPGA image...successful.
openbts@j:~$

```

Fonte: A autora.

Para detectar um equipamento de rádio conectado execute o seguinte comando:

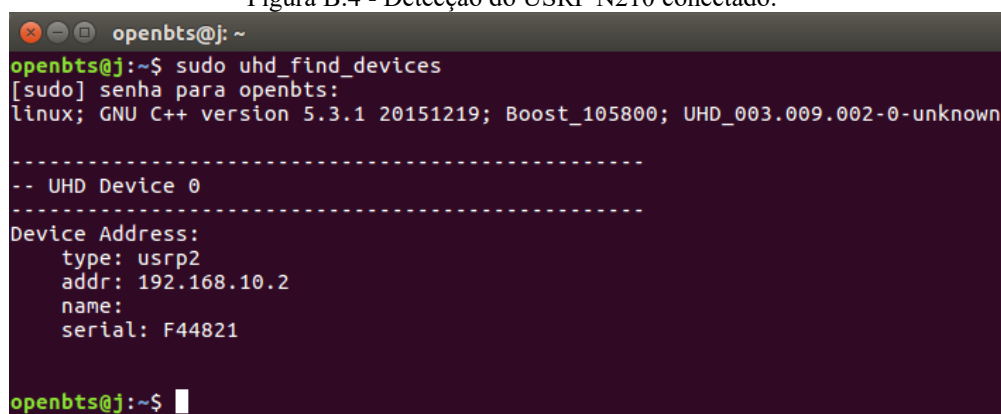
```
$ sudo uhd_find_devices
```

Para inspecionar um equipamento de rádio conectado e retornar suas informações técnicas e configurações execute o comando:

```
$ sudo uhd_usrp_probe
```

As Figuras B.4 e B.5 mostram, respectivamente, a execução dos comandos `uhd_find_devices` e `uhd_usrp_probe` para o USRP N210.

Figura B.4 - Detecção do USRP N210 conectado.



```

openbts@j: ~
openbts@j:~$ sudo uhd_find_devices
[sudo] senha para openbts:
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_003.009.002-0-unknown

-----
-- UHD Device 0
-----
Device Address:
  type: usrp2
  addr: 192.168.10.2
  name:
  serial: F44821

openbts@j:~$

```

Fonte: A autora.

Figura B.5 - Informações técnicas e configurações do USRP N210.

```

openbts@j:~$ sudo uhd_usrp_probe
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_003.009.002-0-unknown

-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes
-- Detecting internal GPSDO.... No GPSDO found

Device: USRP2 / N-Series Device

Mboard: N210r4
hardware: 2577
mac-addr: 00:80:2f:0a:e0:5b
ip-addr: 192.168.10.2
subnet: 255.255.255.255
gateway: 255.255.255.255
gpsdo: none
serial: F44821
FW Version: 12.4
FPGA Version: 11.1

Time sources: none, external, _external_, mimo
Clock sources: internal, external, mimo
Sensors: mimo_locked, ref_locked

RX DSP: 0
Freq range: -50.000 to 50.000 MHz

RX DSP: 1
Freq range: -50.000 to 50.000 MHz

RX Dboard: A
ID: WBX v3, WBX v3 + Simple GDB (0x0057)
Serial: F4419B

RX Frontend: 0
Name: WBXv3 RX+GDB
Antennas: TX/RX, CAL
Sensors: lo_locked
Freq range: 68.750 to 2200.000 MHz
Gain range PGA0: 0.0 to 31.5 step 0.5 dB
Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
Connection Type: IQ
Uses LO offset: No

RX Codec: A
Name: ads62p44
Gain range digital: 0.0 to 6.0 step 0.5 dB
Gain range fine: 0.0 to 0.5 step 0.1 dB

TX DSP: 0
Freq range: -50.000 to 50.000 MHz

TX Dboard: A
ID: WBX v3 (0x0056)
Serial: F4419B
ID: WBX + Simple GDB, WBX v3 + Simple GDB, WBX v4 + Simple GDB, WBX-12
Serial: F43430

TX Frontend: 0
Name: WBXv3 TX+GDB
Antennas: TX/RX, CAL
Sensors: lo_locked
Freq range: 68.750 to 2200.000 MHz
Gain range PGA0: 0.0 to 31.0 step 1.0 dB
Bandwidth range: 40000000.0 to 40000000.0 step 0.0 Hz
Connection Type: IQ
Uses LO offset: No

TX Codec: A
Name: ad9777
Gain Elements: None

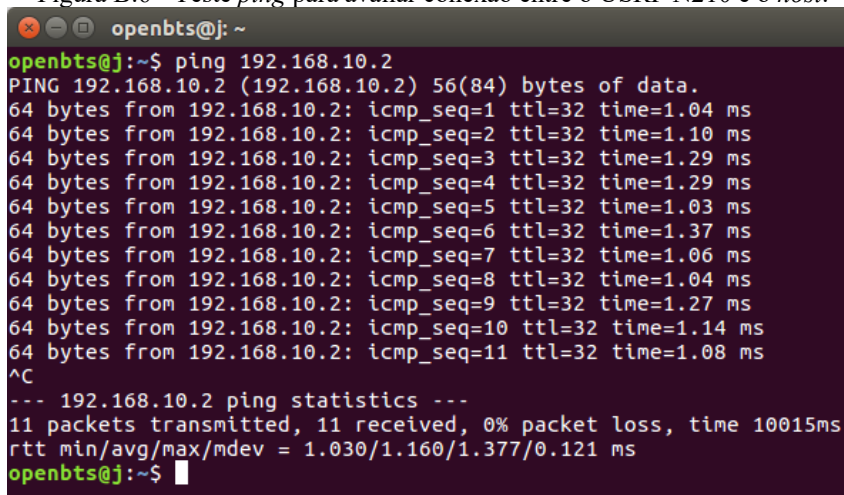
```

Fonte: A autora.

Execute o comando a seguir para testar a conectividade entre o USRP N210 e o *host* (Figura B.6). Para parar este teste pressione as teclas *Ctrl* + *C*.

```
$ ping 192.168.10.2
```

Figura B.6 - Teste *ping* para avaliar conexão entre o USRP N210 e o *host*.



```
openbts@j: ~
openbts@j:~$ ping 192.168.10.2
PING 192.168.10.2 (192.168.10.2) 56(84) bytes of data.
64 bytes from 192.168.10.2: icmp_seq=1 ttl=32 time=1.04 ms
64 bytes from 192.168.10.2: icmp_seq=2 ttl=32 time=1.10 ms
64 bytes from 192.168.10.2: icmp_seq=3 ttl=32 time=1.29 ms
64 bytes from 192.168.10.2: icmp_seq=4 ttl=32 time=1.29 ms
64 bytes from 192.168.10.2: icmp_seq=5 ttl=32 time=1.03 ms
64 bytes from 192.168.10.2: icmp_seq=6 ttl=32 time=1.37 ms
64 bytes from 192.168.10.2: icmp_seq=7 ttl=32 time=1.06 ms
64 bytes from 192.168.10.2: icmp_seq=8 ttl=32 time=1.04 ms
64 bytes from 192.168.10.2: icmp_seq=9 ttl=32 time=1.27 ms
64 bytes from 192.168.10.2: icmp_seq=10 ttl=32 time=1.14 ms
64 bytes from 192.168.10.2: icmp_seq=11 ttl=32 time=1.08 ms
^C
--- 192.168.10.2 ping statistics ---
11 packets transmitted, 11 received, 0% packet loss, time 10015ms
rtt min/avg/max/mdev = 1.030/1.160/1.377/0.121 ms
openbts@j:~$
```

Fonte: A autora.

É necessário verificar se o aplicativo *transceiver* pode se comunicar com o *hardware* USRP N210. Todos os equipamentos de rádio da *Ettus* utilizam o binário *Transceiver52M*.

Diretamente da raiz do *OpenBTS*, execute o seguinte comando para ver se o USRP N210 se comunica com o *OpenBTS*:

```
$ cd /OpenBTS
$ sudo ./transceiver
```

O aplicativo *transceiver* também pode ser acessado através do diretório “*dev*” e, então ser executado conforme os seguintes comandos:

```
$ cd /dev/openbts/Transceiver52M/
$ ls
$ sudo ./transceiver
```

O aplicativo *transceiver* pode ser parado pressionando as teclas *Ctrl* + *C*.

As Figuras B.7 e B.8 mostram tentativas bem-sucedidas de conexão entre o *OpenBTS* e o USRP N210.

Figura B.7 - Conexão entre o *OpenBTS* e o USRP N210 acessando o *transceiver* pela raiz do “*OpenBTS*”.

```

openbts@j: /OpenBTS
openbts@j:~$ cd /OpenBTS
openbts@j:/OpenBTS$ ls
comp128  OpenBTS  OpenBTSCLI  OpenBTSDo  sipauthserve  smqueue  transceiver
openbts@j:/OpenBTS$ sudo ./transceiver
[sudo] senha para openbts:
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_003.009.002-0-unknown

Using internal frequency reference
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes

```

Fonte: A autora.

Figura B.8 - Conexão entre o *OpenBTS* e o USRP N210 acessando o *transceiver* pelo diretório “*dev*”.

```

openbts@j:~$ cd dev
openbts@j:~/dev$ ls
asterisk      build.sh      common.source  liba53      NodeManager  README.md     state.sh      system-config
asterisk-config clone.sh      create.sh      libcoredumper openbts      rename.sh     subscriberRegistry
BUILDS        CommonLibs    delete.sh     LICENSE     pull.sh      smqueue       switchto.sh
openbts@j:~/dev$ cd openbts/
openbts@j:~/dev/openbts$ ls
aclocal.m4    CLI           config.log     COPYING      GSM           ltmain.sh     NodeManager    SIP
apps          CommonLibs    config.status  ctags.sh    GSMShare      Makefile       package        SMS
AUTHORS       compile       config.sub     debian       INSTALL       Makefile.am    Peering        stamp-h1
autogen.sh    config        configure      depcomp     INSTALLATION  Makefile.common README         tools
build-arch-stamp config.guess   configure.ac   doc          install-sh    Makefile.in    README.APIS.md Transceiver52M
build-indep-stamp config.h       configure-stamp Globals       LEGAL         missing        Scanning       TransceiverRAD1
ChangeLog     config.h.in   Control       GPRS        libtool       NEWS           SGSNGGSN      TRXManager
openbts@j:~/dev/openbts$ cd Transceiver52M/
openbts@j:~/dev/openbts/Transceiver52M$ ls
Complex.h      DummyLoad.cpp  Makefile.in    radioInterface.lo  README           sigProcLib.cpp  Transceiver.o
convert.c       DummyLoad.h    pulseApproximate.m radioInterface.o    README.DFESymbolspaced sigProcLib.h    UHDDDevice.cpp
convert.h       DummyLoad.lo   radioClock.cpp  radioInterfaceResamp.cpp README.Talgorithm sigProcLib.lo   UHDDDevice.lo
convert.lo      DummyLoad.o    radioClock.h    radioInterfaceResamp.lo Resampler.cpp     sigProcLib.o    UHDDDevice.o
convert.o       inband-signaling-usb radioClock.lo   radioInterfaceResamp.o Resampler.h       std_inband.rbf  USRPDevice.cpp
convolve.c      laurent.m      radioClock.o    radioVector.cpp    Resampler.lo      transceiver      USRPDevice.h
convolve.h      libtransceiver.la radioDevice.h   radioVector.h      Resampler.o       Transceiver.cpp  Transceiver.h
convolve.lo     Makefile       radioInterface.cpp radioVector.lo     runTransceiver.cpp Transceiver.h    Transceiver.lo
convolve.o      Makefile.am    radioInterface.h radioVector.o      runTransceiver.o
openbts@j:~/dev/openbts/Transceiver52M$ sudo ./transceiver
[sudo] senha para openbts:
linux; GNU C++ version 5.3.1 20151219; Boost_105800; UHD_003.009.002-0-unknown

Using internal frequency reference
-- Opening a USRP2/N-Series device...
-- Current recv frame size: 1472 bytes
-- Current send frame size: 1472 bytes

```

Fonte: A autora.

Referência

[33] USRP2 and N2x0 Series. Disponível em: <
http://files.ettus.com/manual/page_usrp2.html>. Acesso em: 02/06/2018.

APÊNDICE C – Comandos para inicialização do *software* OpenBTS e seus componentes

Após a completa instalação do *software* OpenBTS e seus componentes é necessário testar a inicialização dos componentes. Existem vários comandos para inicializar cada um dos componentes. Cada componente deve ser executado em uma janela separada. A ordem de inicialização não é crítica, porém, se o OpenBTS estiver em execução sem os outros componentes, uma rede GSM ficará visível, mas inutilizável.

O Quadro C.1 apresenta os diversos comandos para inicialização do *Asterisk*, *Sipauthserve*, *Smqueue* e *OpenBTS*.

Quadro C.1 - Comandos para inicialização do OpenBTS e seus componentes.

<i>Asterisk</i>	<i>Sipauthserve</i>	<i>Smqueue</i>	<i>OpenBTS</i>
\$ sudo asterisk	\$ cd /OpenBTS/ \$ sudo ./sipauthserve	\$ cd /OpenBTS/ \$ sudo ./smqueue	\$ cd /OpenBTS/ \$ sudo ./OpenBTS
\$ sudo asterisk -r	\$ sudo su # cd /OpenBTS/ # ./sipauthserve	\$ sudo su # cd /OpenBTS/ # ./smqueue	\$ sudo su # cd /OpenBTS/ # ./OpenBTS
\$ cd /usr/sbin/ \$ sudo asterisk -r	\$ cd dev/subscriberRegistry/apps/ \$ sudo ./sipauthserve	\$ cd dev/smqueue/smqueue/ \$ sudo ./smqueue	\$ cd dev/openbts/apps/ \$ sudo ./OpenBTS
\$ sudo su # cd /usr/sbin/ # asterisk -r	\$ sudo su # cd dev/subscriberRegistry/apps/ # ./sipauthserve	\$ sudo su # cd dev/smqueue/smqueue/ # ./smqueue	\$ sudo su # cd dev/openbts/apps/ # ./OpenBTS

Fonte: IEDEMA, 2015 (Adaptado).

As chaves para configuração do OpenBTS são armazenadas em um banco de dados chamado *SQLite3* que se encontra na rota */etc/OpenBTS/OpenBTS.db*. A manipulação dessas chaves de configuração é feita através da interface de linha de comando do OpenBTS, a *OpenBTSCLI* [2].

Existem as chaves dinâmicas e as chaves estáticas. As chaves dinâmicas podem ter seus valores alterados e serem aplicadas ao sistema em execução em poucos segundos, sem interromper o serviço. Já as chaves estáticas requerem uma reinicialização do OpenBTS para que as mudanças sejam aplicadas [2].

Todos os comandos com prefixo *OpenBTS>* devem ser executados na *OpenBTSCLI*. Já os comandos com prefixo \$ devem ser executados na linha de comandos do *Linux*.

Para acessar a interface *OpenBTSCLI* execute:

```
$ cd /OpenBTS/
$ sudo ./OpenBTSCLI
```

APÊNDICE D – Código para Gráficos

Código desenvolvido no software *Matlab* para construção dos gráficos das Figuras 4.7.10, 4.7.11 e 4.7.12.

```
clear; clc; close all;

% Transceiver
x = [1 2 3 4]; %Número de chamadas
y1 = [35.0 41.3 43.3 46.5]; % %CPU (Delay 4)
y2 = [34.0 39.3 45.5 48.3]; % %CPU (Delay 1)
plot(x,y1,'k',x,y2,'r--')
xlabel ('Número de chamadas')
ylabel ('%CPU para Transceiver')
legend('Delay 4','Delay 1')
grid on

% OpenBTS
figure
x = [1 2 3 4]; %Número de chamadas
y1 = [8.7 12.7 16.0 24.9]; % %CPU (Delay 4)
y2 = [9.0 13.0 16.6 19.3]; % %CPU (Delay 1)
plot(x,y1,'k',x,y2,'r--')
xlabel ('Número de chamadas')
ylabel ('%CPU para OpenBTS')
legend('Delay 4','Delay 1')
grid on

% Asterisk
figure
x = [1 2 3 4]; %Número de chamadas
y1 = [0.3 1.3 2.7 5.3]; % %CPU (Delay 4)
y2 = [0.3 1.7 2.7 4.3]; % %CPU (Delay 1)
plot(x,y1,'k',x,y2,'r--')
xlabel ('Número de chamadas')
ylabel ('%CPU para Asterisk')
legend('Delay 4','Delay 1')
grid on
```