VŠB – Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

OpenBTS a její integrace do telekomunikační sítě

OpenBTS and its Integration into Telecommunication Network

2014 **Ladislav Beháň**

VŠB - Technická univerzita Ostrava
Fakulta elektrotechniky a informatiky
Katedra telekomunikační techniky

Zadání bakalářské práce

Student: **Ladislav Beháň**

Studijní program: B2647 Informační a komunikační technologie

Studijní obor: 2612R059 Mobilní technologie

Téma: OpenBTS a její integrace do telekomunikační sítě
OpenBTS and its Integration into Telecommunication Network

Zásady pro vypracování:

1. GSM síť, její struktura a vlastnosti
2. Projekt OSMOCOM (Open Source Mobile Communication)
3. Vývojový kit openBTS s USRP
4. Praktická realizace GSM BTS s openBTS a scénáře použití
5. Zhodnocení dosažených výsledků

Seznam doporučené odborné literatury:

BURGESS D., HARVIND S. *Open BTS Project.* Kestrel Signal Processing, California, 2008.
KEMETMULLER Ch., SEEGER M., BAIER H., BUSCH Ch. *Manipulating Mobile Devices with a private GSM Base Station - a Case Study.* In Proceedings 8th International Network Conference (INC 2010), 2010.
GUNNAR H. *GSM Networks, Protocols and Implementation.* Artech House, 471 str., 1999, ISBN 0-89006 -471-7

Formální náležitosti a rozsah bakalářské práce stanoví pokyny pro vypracování zveřejněné na webových stránkách fakulty.

Vedoucí bakalářské práce: **doc. Ing. Miroslav Vozňák, Ph.D.**

Datum zadání: 16.11.2012
Datum odevzdání: 07.05.2013

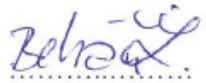
prof. RNDr. Vladimír Vašinek, CSc.
vedoucí katedry

prof. RNDr. Václav Snášel, CSc.
děkan fakulty

Prohlášení studenta

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

Dne: 7.5.2014



podpis studenta

Poděkování

Rád bych poděkoval doc. Ing. Miroslavovi Vozňákovi, Ph.D. za odbornou pomoc a konzultaci při vytváření této bakalářské práce.

Abstrakt

Táto bakalárska práca zoznámi čitateľov so softwarom OpenBTS, prostredníctvom ktorého je spolu s pobočkovou ústredňou Asterisk a softwarovo riadeného vysielača USRP možné vytvoriť si svoju vlastnú GSM sieť. Ďalej detailne popisuje inštaláciu a konfiguráciu jednotlivých komponentov potrebných pre korektnú funkčnosť, ktorá bude overovaná uskutočňovaním telefónnych hovorov a zasielaním sms správ medzi jednotlivými mobilnými zariadeniami v sieti. Ako už z názvu vyplýva, práca sa bude zaoberať možnosťou rozšírenia siete OpenBTS, respektíve jeho pobočkovej ústredne Asterisk, pripojením SIP a IAX trunk providera.

Klíčová slova

GSM, OpenBTS, Asterisk, Smqueue, Subscriber Registry, Osmocom, SIP, IAX, USRP

Abstract

This bachelor thesis will familiarize readers with creating their own GSM network, which can be operated with OpenBTS software, Asterisk PBX and universal software radio peripheral USRP. Further it describes the detail installation and configuration of individual components required for correct functionality, which will be verified by making phone calls and sending text messages between two mobile devices in this GSM network. As the title implies, this thesis will deal with the possibility of linking OpenBTS or more precisely its PBX Asterisk with other provider components, specifically SIP and IAX trunk.

Key words

GSM, OpenBTS, Asterisk, Smqueue, Subscriber Registry, Osmocom, SIP, IAX, USRP

Zoznam použitých skratiek

Skratka	Anglický názov	Slovenský názov
ADC	Administrative Centre	Administratívne centrum
ARFCN	Absolute Radio Frequency Channel Number	Absolútne číslo rádiofrekvenčného kanálu
AuC	Authentication Centre	Centrum autentičnosti
BCH	Broadcast Channels	Rozhlasové kanály
BSC	Base Station Controller	Základňová riadiaca jednotka
BSS	Base Station Sub-System	Subsystém základňových staníc
BTS	Base Tranceiver Station	Základnová stanice
CC	Call Control	Riadenie hovoru
CCCH	Common Control Channels	Kanály všeobecného riadenia
CCH	Control Channels	Riadiace kanály
DCA	Dynamic Channel Allocation	Dynamické prideľovanie kanálov
DCCH	Dedicated Control Channels	Vyhradené riadiace kanály
EIR	Equipment Identity Register	Register mobilních staníc
ETSI	European Technical Standards Institute	Evropský telekomunikační normalizační institut
FCA	Fixed Channel Allocation	Fixné prideľovanie kanálov
FCCH	Frequency Correction Channel	Kanál korekcie kmitočtu
FEC	Forward Error Correction	Dopredná oprava chýb
GMSC	Gateway Mobile Swiching Center	Brána mobilnej ústredne
GSM	Global System for Mobile Communication	Globálny systém pre mobilnú komunikáciu
HLR	Home Location Register	Databáza domovských účastníkov
IMEI	International Mobile Equipment Identity	Výrobní číslo telefónu
IMSI	International Mobile Subscriber Identity	Medzinárodní identifikační číslo
IWF	Inter-Working Functionality	Jednotka spolupráce
LAI	Local Area Identification	Kód lokalizačnej oblasti
MM	Mobility Management	Managment mobility
MS	Mobile Station	Mobilná stanica
MSC	Mobile Switching Center	Mobilná rádiotelefónna ústredňa
NMC	Network Monitoring Center	Centrum managementu siete
NSS	Network Switching Subsystem	Sieťový spojovací subsystém
OMC	Operational and Maintenance Centre	Prevádzkové a servisné centrum

OSS	Operational and Support Subsystem	Operačný a podporný subsystém
PIN	Personal Identification Number	Osobné identifikačné číslo
RACH	Random Access Channel	Kanál náhodného prístupu
RAM	Random Access Memory	Pamäť s nádodným prístupom
ROM	Read-Only Memory	Pamäť iba pre čítanie
RR	Radio Resource	Management rádiových zdrojov
SACCH	Slow Asscociated Control Channel	Pomalý pridružený riadiaci kanál
SDCCH	Stand Alone Dedicated Control Channel	Samostatný pridelený riadiaci kanál
SDR	Software Defined Radio	Softwarovo definované rádio
SIP	Session Initiation Protocol	Protokol pre incializáciu relácií
SMSC	Short Message Service Center	Stredisko krátkych textových správ
TCH	Traffic Channels	Prevádzkové kanály
TDM	Time Division Multiplexing	Viacnásobný prístup s časovým delením
TMSI	Temporary Mobile Subscriber Identity	Dočasná identifikácia mobilného účastníka
TRAU	Transcoder and Rate Adaptation Unit	Transkódovacia jednotka
TRX	Transceiver	Vysielač
USRP	Universal Software Radio Peripheral	Univerzálny softwarovo riadený
UHD	USRP Hardware Driver	USRP hardwarový ovládač
VLR	Visitor Location Register	Databáza hosťujúcich účastníkov
VMS	Voice Mail Systém	Záznamová služba
VoIP	Voice over Internet Protocol	Internetová telefónia

Obsah

1 Úvod ... 1

2 GSM sieť, jej štruktúra a vlastnosti ... 2

2.1 Celulárna štruktúra GSM ... 2

2.2 Rozloženie kmitočtového pásma ... 3

2.3 Architektúra GSM ... 4

2.3.1 Mobilná stanica MS ... 4

2.3.2 Subsystém základňových staníc BSS ... 5

2.3.3 Sieťový spojovací subsystém NSS ... 7

2.3.4 Operačný a podporný subsystém OSS ... 9

2.4 Vrstvový model GSM ... 9

2.4.1 Fyzická vrstva ... 9

2.4.2 Spojová vrstva ... 11

2.4.3 Sieťová vrstva ... 11

3 Projekt OSMOCOM ... 12

3.1 OpenBSC ... 12

3.2 OsmoBTS ... 13

3.3 Libosmocore/Libosmo-Abis ... 13

3.4 OsmoSGSN/OpenGGSN ... 13

3.5 Osmo-PCU ... 13

3.6 Komerčná BTS a OpenBSC ... 14

4 Vývojový kit openBTS s USRP ... 15

4.1 Úvod do OpenBTS ... 15

4.2 Architektúra OpenBTS ... 15

4.2.1 Vrstvový model OpenBTS: ... 16

4.2.2 Hardwarové požiadavky ... 16

4.3 USRP ... 17

4.4 USRP Hardware Driver UHD ... 18

4.5 Smqueue ... 18

4.6 Subscriber registry ... 18

4.7 Asterisk ... 18

4.7.1 SIP protokol ... 19

4.7.2 IAX protokol ... 19

5 Praktická realizácia GSM BTS s openBTS a scenáre použitia 21
5.1 Inštalácia OpenBTS 21
5.2 Inštalácia smqueue 22
5.3 Inštalácia sipauthserve a Subscriber Registry 22
5.4 Inštalácia a konfigurácia Asterisku 23
5.5 Spustenie 25
5.6 Scenáre použitia 28
5.6.1 Konfigurácia spojenia mobilných staníc v sieti OpenBTS 28
5.6.2 Pripojenie na SIP trunk providera 32
5.6.3 Zabezpečenie SIP trunku prostredníctvom SIP TLS 35
5.6.4 Pripojenie na IAX trunk providera 37
6 Zhodnotenie dosiahnutých výsledkov 40
6.1 Realizácia hovoru a posielania správ v sieti OpenBTS 40
6.2 Realizácia hovoru pripojením na SIP trunk providera 41
6.3 Reliazácia hovoru pripojením na IAX trunk providera 43
7 Záver 45
8 Použitá literatura 46
Zoznam príloh 48

Zoznam tabuliek:

Tabuľka 2.1: Úlohy VF a NF časti BTS stanice 6

Tabuľka 2.2: Popis logických kanálov 10

Zoznam obrázkov:

Obrázok 1: Buňkový systém GSM ... 2

Obrázok 2: Architektúra GSM ... 4

Obrázok 3: Schéma BTS stanice ... 5

Obrázok 4: Schéma NSS ... 8

Obrázok 5: Vrstvový model GSM ... 9

Obrázok 6: Architektúra OpenBTS ... 16

Obrázok 7: USRP N210 ... 17

Obrázok 8: Schéma zapojenia ... 25

Obrázok 9: No UHD Devices Found ... 26

Obrázok 10: GUI pre aktualizáciu USRP ... 26

Obrázok 11: Uvítacia správa ... 28

Obrázok 12: Registrácia IMSI ... 29

Obrázok 13: dial_data_table ... 30

Obrázok 14: sip_buddies_table ... 30

Obrázok 15: Pripojenie na SIP trunk providera ... 32

Obrázok 16: Konfigurácia Yate ... 35

Obrázok 17: SIP TLS (Wireshark) ... 37

Obrázok 18: Pripojenie na IAX trunk providera ... 37

Obrázok 19: SMS ... 40

Obrázok 20: Tel.hovor Wireshark ... 41

Obrázok 21: SMS Wireshark ... 41

Obrázok 22: Tel.hovor Wireshark (SIP trunk) ... 42

Obrázok 23: softphone YATE (tel.č 2222) ... 42

Obrázok 24: IAX trunk Wireshark ... 43

Obrázok 25: Tel.hovor Wireshark (IAX trunk) ... 44

1 Úvod

V súčasnosti však GSM čoraz viac priťahuje pozornosť open-source komunity. Softwarová implementácia tradičnej GSM siete by umožnila jej prevádzku za omnoho nižšie náklady, lepšie prispôsobenie potrebám uživateľa a zabezpečila by jednoduchšiu kontrolu nad celým systémom. V tejto bakalárskej práci sa budem venovať práve dvom projektom, ktoré umožňujú uživateľovi na základe softwarovej implementácie súčasnej GSM siete, si vytvoriť vlastnú rôznokapacitnú mobilnú sieť v závislosti na jeho individuálnych potrebách. Týmito projektami sú Osmocom a hlavne OpenBTS.

Prostredníctvom použitia projektu Osmocom spolu s kompatibilným telefónom je možné vytvárať a príjmať telefónne hovory, či posielať a príjmať textové správy. Osmocom pre integráciu do telekomunikačnej siete potrebuje pripojenie na klasickú základňovú stanicu (BTS), čo je nevýhodné vzhľadom na jej obmedzenú dostupnosť a neprístupný zdrojový kód.

Hlavnou témou mojej bakalárskej práce je projekt OpenBTS, ktorý poskytuje vytvorenie vlastnej GSM siete s minimálnou funkcionalitou prostredníctvom lacného a ľahko dostupného univerzálneho softwarovo definovaného vysielača (USRP). Táto open-source Unixová aplikácia umožňuje ktorémukoľvek užívateľovi sprevádzkovať mobilnú sieť za desatinu prevádzkových nákladov terajšej GSM siete. Vďaka ekonomickému softwarovému dizajnu a patričným výberom výkonných zosilovačov by malo byť možné spustiť nízkokapacitnú GSM buňku zo solárnych panelov alebo prostredníctvom veterných turbín, čím by bola dosiahnutá oveľa menšia spotreba energie než v prípade súčasnej technológie. Tento projekt využíva pre smerovanie hovorov voľne stiahnuteľnú Unixovú aplikáciu Asterisk, ktorý na prostredníctvom viacerých protokolov, umožňuje pripojenie k rozličným privátnym a verejným IP sieťam. Preto využitím projektu OpenBTS by si mohli užívatelia po celom svete navzájom telefonovať prostredníctvom internetu.

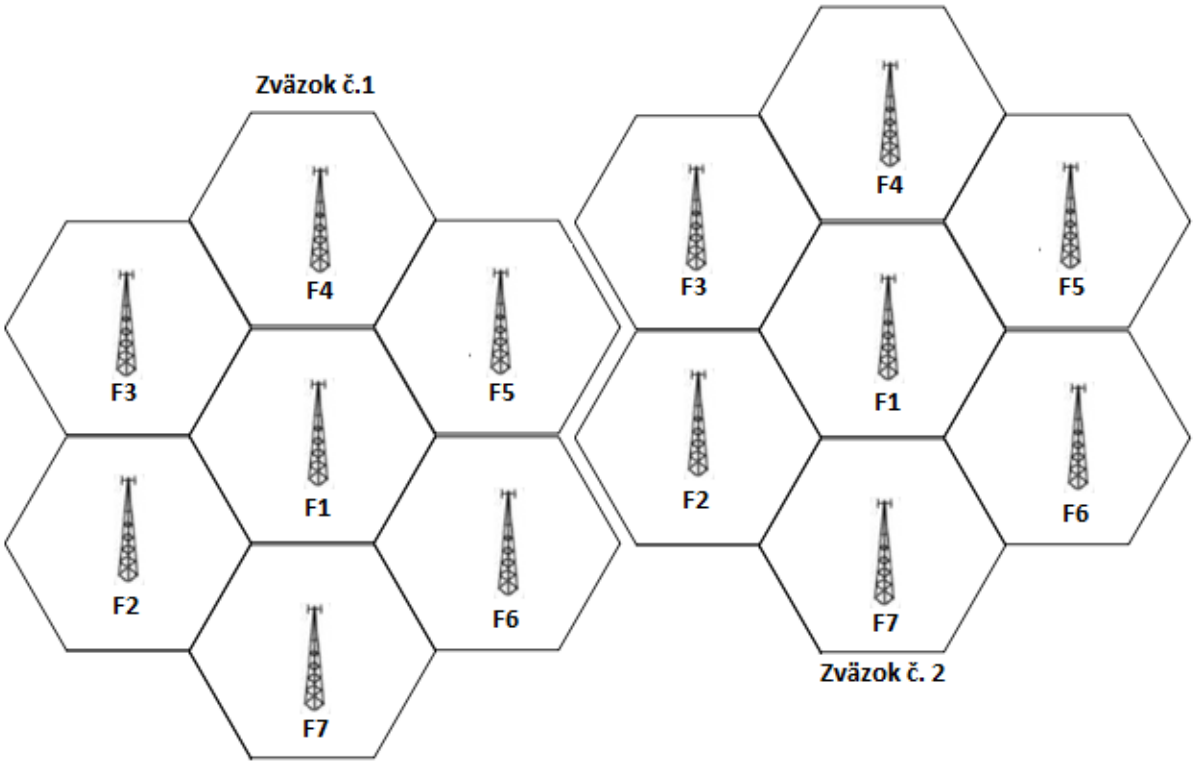
Celá bakalárska práca je rozdelená do piatich hlavných kapitol. V prvej kapitole sa venujem tradičnej GSM sieti, jej štruktúre, funkciám a protokolovým vrstvám. Druhá časť je venovaná projektu Osmocom, priblížim princíp funkčnosti tohoto projektu a popíšem jednotlivé komponenty, z ktorých pozostáva. Tretia kapitola popisuje open-source projekt OpenBTS, jeho ciele, architektúru a ktoré vrstvy GSM siete implementuje. V následujúcej kapitole priblížim všetky potrebné náležitosti k realizácii vlastnej GSM siete prostredníctvom OpenBTS, zároveň rozšírim túto sieť o pripojenie na SIP alebo IAX trunk providera. V záverečnej kapitole budem demonštrovať funkčnosť systému prostredníctvom zasielanie textových správ a uskutočňovania telefónnych hovorov medzi telefónnymi zariadeniamu. Celú komunikáciu budem zachytávať prostredníctvom protokolového analyzéru, aplikácie Wireshark

2 GSM sieť, jej štruktúra a vlastnosti

V tejto časti bakalárskej práce sa budeme venovať súčasnej GSM sieti a jej architektúre. Priblížime si jednotlivé komponenty, z ktorých pozostáva a zároveň sa bližšie pozrieme bližšie na vrstvový model GSM. V tejto kapitole je čerpané z následujúcich informačných zdrojov: [1], [2], [3], [4], [5], [6].

2.1 Celulárna štruktúra GSM

Každý mobilný operátor môže získať od štátneho orgánu, ktorý má na starosti správu frekvenčného spektra, len obmedzené frekvencie pre fungovanie mobilnej siete. Rozsahy frekvencií, ktoré boli pridelené operátorom sa môžu podstatne odlišovať, avšak na to aby bol pre každý hovor v sieti vymedzený vlastný kanál nepostačujú. Riešením problému s nedostatkom voľných frekvencií, je mnohonásobné využitie rovnakého prideleného kmitočtu, aby rôzne hovory využívali v rovnaký čas rovnaký kmitočet. Preto sa v súčasnosti využíva celulárny (buňkový) systém.



Obrázok 1: *Buňkový systém GSM*

Vďaka tomuto efektívnemu spôsobu hospodárenia s frekvenčným spektrom, môže takpovediac neobmedzenému počtu účastníkov systému slúžiť malý počet rádiových kanálov. Príklad realizácie buňkového systému ukazuje obrázok 1. Na obrázku je znázornená mobilným operátorom

obsluhovaná oblasť, ktorá je rozdelená na dva zväzky, ktoré obsahujú po 7 rovnako veľkých, šesťuholníkových buniek.

Veľkosť jednotlivých buniek je určená ich priemerom R, ktorý predstavuje vzdialenosť stredov dvoch susedných buniek. Na základe priemeru, na začiatku využívania celulárneho systému, sa rozlišovali pikobuňky s priemerom menším ako 100 metrov, mikrobuňky (R = 100 – 300 m), malé buňky (R = 300 – 3000 m), stredné buňky (R = 3000 – 10000 m) a veľké buňky (R = 10000 – 30000 m).

Každá buňka obsahuje základňovú rádiovú stanicu BTS (Base Tranceiver Station), ktorá predstavuje vysielač a príjmač rádiových signálov s osobitnou skupinou kanálov. Vo vnútri nachádzajúci sa mobilný účastník, pomocou tejto základňovej stanice, nadväzuje spojenie s okolitým svetom. Ostatné buňky celulárnej štruktúry majú taktiež vlastné skupiny kanálov. Jednotlivé buňky sú usporiadané tak, že priamo susediace buňky nepoužívajú zhodné frekvencie.

Pokiaľ buňkovému systému bude pevne pridelený určitý počet kanálov, môžeme ich rozdeliť tak, že prvá buňka povedzme môže zahrňovať kanály č. 1 až 111, druhá buňka bude obsahovať kanály č. 112 až 222, tretia rádiové kanály č. 223 až 333, a tak to pôjde ďalej až po siedmu poslednú buňku, ktorá bude pozostávať z kanálov č. 667 až 777. Celý zväzok buniek je teda zložený zo 777 rádiových kanálov. Druhý zväzok, priamo susediaci s prvým zväzkom, môže použiť rovnaký počet kanálov. Vzdialenosť dvoch buniek z rôznych zväzkov, ktoré využívajú súhlasné kanály, ako je zobrazené na obr.1, musí byť rovná približne päťnásobku polomeru buňky, aby nedochádzalo k vzájomnému rušeniu rádiovej prevádzky v týchto bunkách. Pokiaľ by sme takýmto spôsobom pridali ďalšie zväzky s identickou konfiguráciou buniek a rádiových kanálov, mohli by sme so 777 rádiovými kanálmi pokryť územie celého sveta. Tento spôsob prideľovania kanálov do každej buňky sa nazýva pevné prideľovanie kanálov a je označený skratkou FCA (Fixed Channel Allocation).

Ďaľšiou metódou prideľovania rádiových kanálov je dynamické prideľovanie, označované aj DCA (Dynamic Channel Allocation), ktorý umožňuje v danej chvíli preťaženej buňke, použiť kanály, ktoré boli prvotne pridelené iným buňkám vo zväzku.

2.2 Rozloženie kmitočtového pásma

V systéme GSM je kmitočtové pásmo v rozsahu 890 - 960 Mhz rozdelené do dvoch častí. Pre spojenie MS s BTS v smere uplink sa využíva pásmo 890 - 915 MHz a pre spojenie od BTS k MS v smere downlink je vyhradené pásmo 935 – 960 MHz. GSM umožňuje fullduplex prevádzku prostredníctvom kmitočtového duplexu FDD s duplexným odstupom 45 Mhz a odstupom 200 Khz medzi nosnými vlnami. PGSM celkovo obsahuje 125 kanálov, ale keďže nultý kanál je oddeľovací a nepoužiteľný pre prenos hovorov, tak celkovo využiteľných duplexných kanálov je 124.

Každému duplexnému kanálu je priradené číslo ARFCN (Absolute Radio Frequency Channel

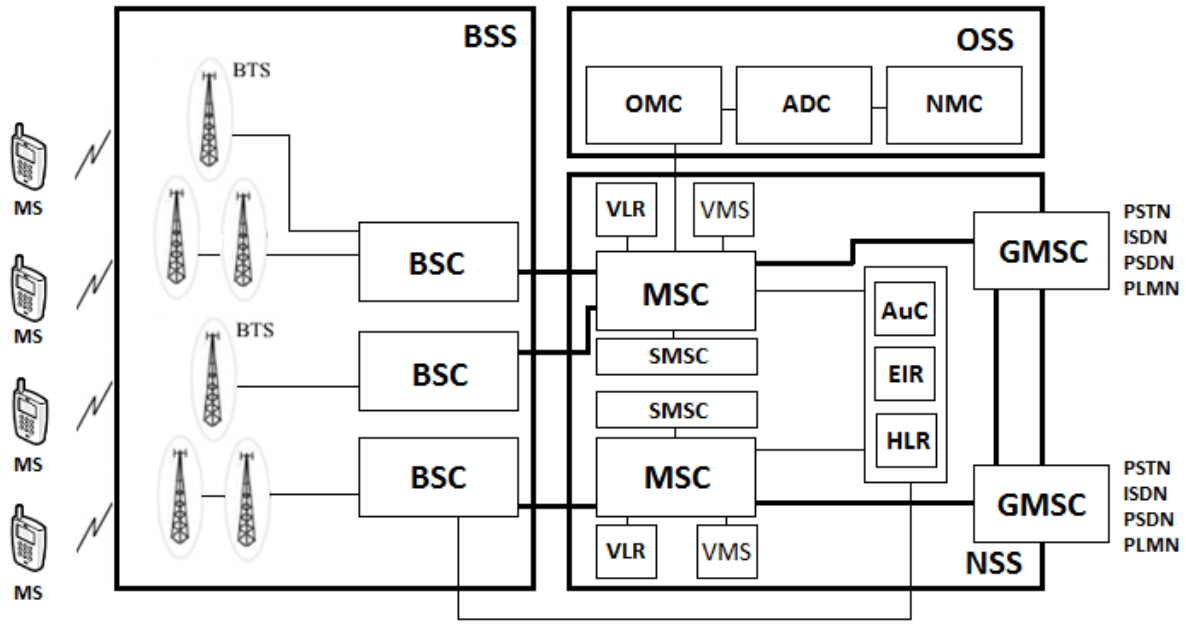
Number), pre identifikáciu jednotlivých frekvenčných kanálov pomocou celého čísla. V GSM sú preto jednotlivých frekvenčným kanálom priradené hodnoty 1-124.

Nakoľko kapacita v oblasti 900 Mhz bola nedostatočná, preto vznikol systém GSM 1800 so širkou pásma 75 MHz a duplexným odstupom 95 Mhz, ktorý sa líši vo využívanom kmitočtovom pásme (1710 – 1785/1805 - 1880) a poskytuje až 375 rádiových kanálov s číslom ARFCN od 512 po 885.

V rozšírenom systéme EGSM je frekvenčné pásmo rozšírené na 880 až 915 Mhz v smere uplink a vo smere downlink to je 925 až 960 Mhz. Týmto rozšírením PGSM o 10 Mhz na spodných okrajoch sa zvýšila kapacita systému o 50 duplexných kanálov (ARFCN 975 až 1023).

2.3 Architektúra GSM

Celú GSM sieť môžme rozdeliť do troch subsystémov(obr. 2):



Obrázok 2: *Architektúra GSM*

2.3.1 Mobilná stanica MS

Mobilnou stanicou sa rozumie GSM telefón, ktorý sa prostredníctvom Um rozhrania pripojí k BTS stanici, alebo SIM karta zásluhou ktorej je každý mobilný účastník v sieti identifikovaný. Každá mobilná stanica obsahuje unikátne jednoznačné identifikačné číslo (IMEI), ktoré sa využíva napríklad pre zablokovanie zariadenia, pri jeho prípadnom odcudzení. Pokiaľ MS nemá vloženú SIM kartu, je v rámci GSM siete nepoužiteľný a môže uskutočňovať iba núdzové hovory.

SIM obsahuje čip s mikroprocesorom a prostredníctvom pamätí RAM a ROM ukladá užívateľské dáta, ktorými sú jednoznačná identifikácia užívateľa (IMSI), štvormiestny číselný PIN kód, PUK kód, roamingové dáta, telefónne čísla, SMS správy, dočasná identifikácia mobilného účastníka (TMSI) a číselný kód oblasti (LAI).

Najdôležitejšími funkciami MS sú:

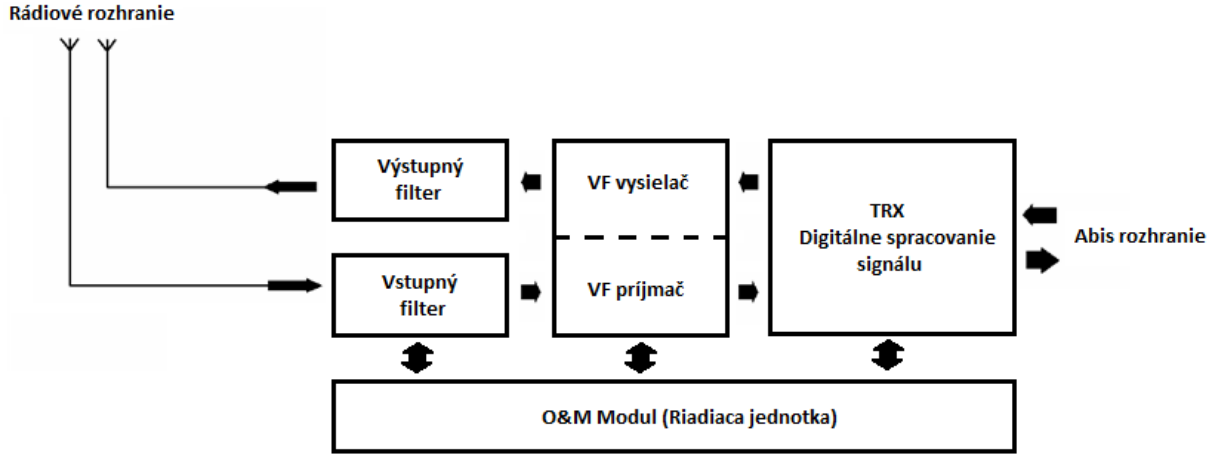
- kódovanie a dekódovanie signálov
- zaistenie synchronizácie a ekvalizácie
- prenos hovoru a dát
- analýza kvality signálu v okolitých buňkách z dôvodu optimalizácie handoveru
- príjem a zobrazenie SMS správ

2.3.2 Subsystém základňových staníc BSS

Subsystém BBS cez rádiové Um rozhranie zabezpečuje spojenie s mobilnými stanicami a je zložený z nasledujúcich prvkov:

- ***Základňová stanica BTS***

BTS poskytuje fyzické spojenie mobilných staníc vo forme rádiového rozhrania a smerom k NSS, BSC je pripojená k BTS skrz Abis rozhranie. Základňová stanica môže podľa GSM doporučení pojať 16 modulov TRX(vysielač/príjmač), avšak vo väčšine prípadov BTS stanica obsahuje maximálne štyri TRX moduly. Schéma BTS stanice obsahujúcej práve jeden TRX modul, môžeme vidieť na obrázku 3.



Obrázok 3: *Schéma BTS stanice*

BTS sa stanica sa skladá z:

- TRX modulu, ktorý pre spracovanie digitálnych signálov využíva svoju nízkofrekvenčnú časť a pre GMSK moduláciu alebo demoduláciu, časť vysokofrekvenčnú. Tieto dve frekvenčné

časti sú spojené pomocou jednotky frekvenčného skákania, ktorá je buď separátna alebo spoločná. Tabuľka 1 pokukazuje na to, ktoré úlohy plní nízkofrekvenčná časť (NF) a ktoré pre zmenu vysokofrekvenčná(VF).

Funkcie	**NF**	**VF**
Kanálové kódovanie a dekódovanie	✓	
Prekládanie a spätné preusporiadanie	✓	
Šifrovanie a dešifrovanie	✓	
Pomalé frekvenčné skákanie	✓	
Formátovanie burstov	✓	
TRAU formátovanie rámcov a konverzia od/do BSC, zriadenie LAPD spojenia s BSC	✓	
GMSK modulácia downlink dát	✓	✓
GMSK demodulácia všetkých prijatých signálov od MS	✓	✓
Tvorba a prenos BCCH na nultý kanál BCCH-TCH	✓	✓
Meranie sily a kvality signálu pre aktívne spojenia	✓	✓
Poskytnutie výsledkov pre BSC	✓	✓
Merania interferencie na voľných kanálov a zaslanie výsledkov do BSC	✓	✓

Tabulka 1: Úlohy VF a NF časti BTS stanice[5]

- O&M modulu, ktorý pozostáva prinajmenšiom z jednej centrálnej jednotky a jej úlohou je riadenie zvyšných častí BTS. Z tohto dôvodu je BTS priamo pripojená na BSC prostredníctvom O&M kanálu. Vďaka O&M kanálu môžu BTS stanice priamo spracovávať príkazy z BSC a MSC.

- Modulu časovania, z dôvodu testovania BTS pokiaľ nie je pripojená k BSC, alebo ak nie sú k dipozícii PCM hodiny. Časovanie v GSM sieti je nevyhnuteľné, pretože všetky TRX základňovej stanice musia používať rovnaký hodinový signál s presnosťou 0.05 ppm.

- Vstupné a výstupné filtre, ktoré sa využívajú pre obmedzenie šírky pásma či už pre prijaté signály alebo signály vysielané. Vstupné filtre dovolia v smere uplink prejsť frekvenciám GSM 900, 1800 a 1900. V smere downlink sa využíva výstupný filter, ktorý obmedzuje šírku pásma pre výstupné signály na 200 kHz. Výstupné filtre sú diaľkovo ovládané a v prípade nutnosti aj prenastaviteľné pomocou Administratívnej jednotky.

- ***Base Station Controller BSC***

Základňová riadiaca jednotka BSC tvorí stred BSS. Prostredníctvom rozhrania Abis môže spojovať mnoho BTS staníc, v rámci komunikácie BTS s MS prideľovať a uvoľňovať rádiové kanály a taktiež zabezpečuje handover mobilných staníc. BSC aplikuje dynamické prideľovanie kanálov, čím na rozdiel od fixného prideľovania, docieli zreteľne vyššiu prevádzkovú kapacitu. BSC komunikuje s MSC prostredníctvom A rohrania a s BTS pomocou rozhrania Abis.

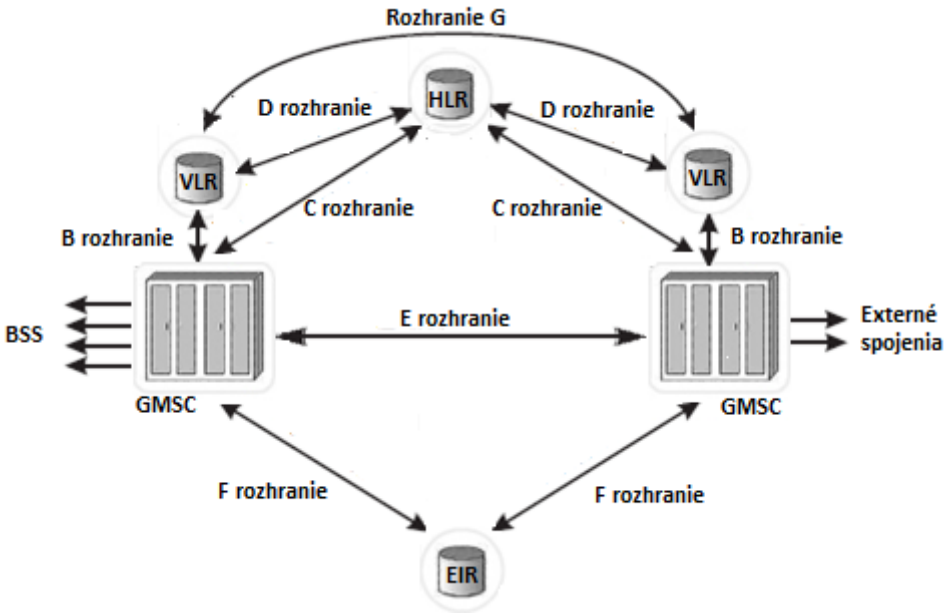
2.3.3 Sieťový spojovací subsystém NSS

NSS plní prepojovaciu funkciu a spája externé komunikačné siete so sieťou GSM. Tento subsystém je prepojený so subsystémom BSS na jednej strane a na druhej s už spomínanými externými sieťami. Najhlavnejším prvkom subsystému je mobilné prepojovacie centrum MSC (Mobile Switching Center), ktoré principiálne funguje ako klasická telefónna ústredňa. Roaming, autentizácia, šifrovanie či lokalizačné služby sú ďalšie funkcie NSS a pre ich vykonanie pozostáva z nasledujúcich častí:

- **Mobile Switching Center MSC -** V subsystéme NSS uskutočňuje prepojovacie funkcie mobilná rádiotelefónna ústredňa MSC. Medzi funkcie MSC patrí aj riadenie prevádzky medzi všetkými BSC, ktoré podliehajú danej ústredni. Táto ústredňa spája mobilnú sieť s rozlišnými typmi pevných či mobilných sietí. Prostredníctvom prepojovacej funkcie IWF (Interworking Function) , vykonáva prepojenie na iné siete napr. ISDN, PDN či PSTN a sleduje pohyb mobilných staníc.

- **Gateway Mobile Switching Center GMSC -** Rozhranie medzi mobilnou sieťou a sieťami vonkajšími, tvorí ďalšia časť subsystému NSS a to je brána mobilnej ústredne GMSC. MSC, ktoré nedisponuje funkciou brány, musí smerovať hovory do externých sietí práve prostredníctvom brány GMSC.

- **Databáza domovských účastníkov HLR** - Všetky podstatné informácie o mobilných účastníkoch v oblasti ústredne NSS, sú uložené v tejto databáze. HLR obsahuje napr. IMSI čísla, volacie číslo MS, dostupné doplňkové služby, informácie o aktuálnej polohe účastníka apod. Každý účastník je uložený v práve jednej HLR databázi a to z dôvodu prevencie pred možnými chybami spôsobenými zastaralými údajmi uloženými v inej HLR databázi.



Obrázok 4: *Schéma NSS*

- **Databáza hosťujúcich účastníkov VLR** - Nasledujúcou databázou a súčasťou NSS je databáza hosťujúcich účastníkov VLR a obsahuje informácie súvisiace s polohou mobilného účastníka nachádzajúceho sa v prevádzkovej oblasti NSS. Akonáhle sa účastník dostane do tejto oblasti, jeho IMSI a informácie o doplňkových službách sú skopírované do VLR a ten si vzhľadom k IMSI vytvorí dočasné identifikačné číslo TMSI a pripojí kód aktuálnej lokalizačnej oblasti LAI (Local Area Identification.) Tieto údaje sú vo VLR uložené dočasne a v okamžiku, kedy účastník opustí danú oblasť, sú z tejto databáze vymazané.

- **Databáza centra overovania AuC** - Jedná sa o chránenú databázu, ktorá obsahuje autentizačné kľúče pre jednotlivých účastníkov uložených v registroch HLR a VLR, A3 algoritmus pre overenie totožnosti účastníka a A8 algoritmus pre generovanie šifrovacieho kľúča, pomocou ktorého je každému účastníkovi pridelený jedinečný časovo označený kľúč, čo zvyšuje bezpečnosť v GSM sieti.

- **Databáza zariadení EIR** - V EIR sú na základe medzinárodných identifikačných čísiel IMEI zaznamenané informácie o povolených, stratených alebo ukradnutých mobilných staniciach. Každý mobilný operátor má v celej sieti len jednu EIR databázu. Mobilní operátori môžu rozhodnúť o použití IMEI pre kontrolu pri zostavovaní spojenia. Pri kontrole najskôr MSC-VLR vyžiada od mobilnej stanice jeho IMEI a prijatá identifikácia je následne uložená do databáze EIR, ktorá ju skontroluje a začlení do jedného z nasledujúcich zoznamov:
    - **„Biely“ zoznam** obsahujúci IMEI platne registrovaných MS

- **„Čierny“ zoznam** obsahujúci IMEI nahlásených ukradnutých MS
- **„Šedý“ zoznam** obsahujúci IMEI poškodených MS

2.3.4 Operačný a podporný subsystém OSS

OSS zabezpečuje nasledujúce základné funkcie:

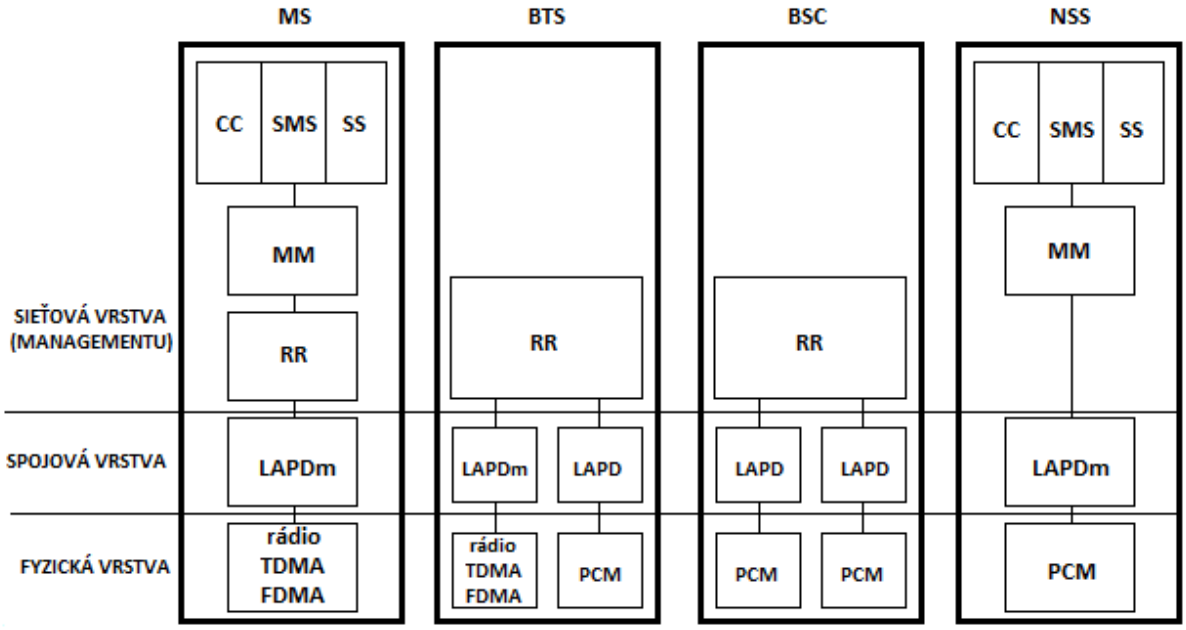
- riadi prevádzku a stará sa o údržbu hardwaru BSS a NSS
- sleduje účastnícku registráciu a sčasti rieši tarifikáciu
- zabezpečuje monitoring mobilných staníc a zisťuje pokazené stanice

Spomenuté úlohy plnia v subsystéme OSS nasledujúce komponenty:

- prevádzkové a servisné centrum OMC
- administratívne centrum ADC
- centrum managmentu siete NMC

2.4 Vrstvový model GSM

Na obrázku 5 je graficky znázornený vrstvový model systému GSM, definujúci tri najspodnejšie vrstvy referenčného modelu OSI. Jednotlivé vrstvy obsahujú skupinu entit, ktoré predstavujú ich funkčné jednotky.



Obrázok 5: *Vrstvový model GSM*

2.4.1 Fyzická vrstva

GSM využíva viacnásobný prístup frekvenčným delením FDMA pre rozdelenie širších frekvenčných pásiem do menších 200 kHz kanálov.

Okrem FDMA, GSM taktiež využíva viacnásobný prístup časovým delením TDMA, ktorý rozdeľuje jednotlivé rádiové kanály do ôsmich časových slotov, čo umožňuje ôsmim uživateľom využívať jedinú frekvenciu. Doba trvania každého časového slotu je 576.9 mikrosekúnd, z toho vyplýva, že TDMA rámec, pozostávajúci z ôsmich časových slotov, bude mať dobu trvania 4.615 milisekúnd.

Multirámec signalizačného kanálu (CCH) pozostáva z 51 TDMA rámcov a multirámec prevádzkového kanálu (TCH) je zložený z 26 TDMA rámcov. Superrámec CCH vznikne zložením 26 CCH rámcov a na rozdiel od TCH superrámca, ktorý vznikne spojením 51 TCH multirámcov. V konečnom dôsledku superrámec obidvoch kanálov pozostáva celkovo z 1326 TDMA rámcov.

Signalizačné kanály rozdeľujeme do troch kategórii. Prvou je rozhlasový kanál (BCH) s funkciou jednosmerného vysielania. Druhú kategóriu tvoria kanály všeobecného riadenia (CCCH), ktoré umožňujú funkciu pre bod-mnohobod prístup. Ďalšiou skupinou sú vyhradené riadiace kanály (DCCH) s funkciami pre obojsmerné spojenie typu bod-bod. Okrem týchto logických kanálov sa v sieti GSM nachádzajú prevádzkové kanály, ktoré sú určené pre realizáciu prenosu digitalizovaných hovorov a dátových signálov. V tabuľke 2 sú stručne popísané jednotlivé logické kanály.

Kanál	Typ	Popis
FCCH	**BCH**	prenos nemodulovaných burstov pre kmitočtovú korekciu (samé nuly
SCH	**BCH**	prenos identifikačného kódu základňovej stanice BSIC, informácie pre rámcovú signalizáciu MS a je tvorený synchronizačným burstom.
BCCH	**BCH**	prenos konfigurácie logického kanálu, frekvencie susedných buniek, synchronizačné informácie
CBCH	**BCH**	prenos textových správ alebo oznámení pre všetky MS.
RACH	**CCCH**	vzostupný kanál, ktorý pokiaľ MS požiada o spojenie, tak príjma prístupové bursty.
AGCH	**CCCH**	MS je priradený riadiaci kanál, o ktorý si predtým zažiadala. Tvorený normálnym burstom.
PCH	**CCCH**	informuje MS o prichádzajúcom hovore a obsahuje normálny burst.
SDCCH	**DCCH**	nesie signalizáciu pre funkcie medzi MS a BTS, ako je napríklad aktualizácia umiestnenia, zostavenie spojenia a taktiež nesie SMS data.
SACCH	**DCCH**	prenos signalizácie pre riadenie výkonu, nesie SMS data počas hovoru MS.
FACCH	**DCCH**	používaný počas komunikácie MS.Prenos signalizácie pre zostavenie hovoru.
TCH	**TCH**	jednosmerný prenos digitalizovaných hovorov a dátových signálov

Tabuľka 2: *Popis logických kanálov*

2.4.2 Spojová vrstva

Spojová vrstva zabezpečuje obalenie dát pre prenos. Tieto údaje sú zlúčované do paketov alebo rámcov a ďalej poskytnuté fyzickej vrstve kvôli synchrónnemu či asynchrónnemu prenosu. Súčasťou spojovej vrstvy je komunikačný protokol HDLC, ktorý svojou činnosťou zabezpečuje detekciu chýb, riadenie toku dát a poskytuje všeobecnú štruktúru dátových rámcov.

Dátový rámec je na začiatku a na konci tvorený krídlovou značkou, ďalej štruktúra rámca tvorí adresné pole, riadiace pole a kontrolný súčet.

Hlavným účelom spojovej vrstvy je detekcia chýb a ich následná oprava. Pokiaľ príjmač detekuje chybu tak sa ju pokúsi napraviť alebo požiada o opakovaný prenos.

Spojová vrstva Um rozhrania je tvorená protokolom LAPDm, spolu s kanálovým kódovaním a formatovaním burstov. LAPDm predstavuje upravenú verziu LAPD z ISDN, ktorý sa nachadza na rozhraní Abis a neobsahuje kontrolný súčet, pretože o detekciu a korekciu chýb sa stará kanálové kódovanie. LAPDm podporuje len dve prístupové miesta siete (SAP) pre pripojenie k službám sieťovej služby a to SAP0 pre RR, MM a CC a SAP3 pre SMS.

2.4.3 Sieťová vrstva

Sieťová vrstva je zodpovedná za smerovanie dátových paketov a delí sa na tri podvrstvy:

- **Management rádiových zdrojov RR** - Managment rádiových zdrojov je najspodnejšiou podvrstvou sieťovej vrstvy GSM systému. RR zabezpečuje spojenie medzi MS a BSS, riadi logické a fyzické kanály na Um rozhraní a handover.

- **Management mobility MM -** Podvrstva managment mobility poskytuje služby pre entity podvrstvy CM. Hlavným cieľom je podpora mobility a registrácie MS v celej GSM sieti. Medzi MM procedúry patrí autentizácia, identifikácie, realokácia TMSI , pripojenie a odpojenie IMSI a aktualizáciu polohy.

- **Managament komunikácie CM -** Podvrstva CM sa delí na nasledujúce štyri entity: Riadenie hovoru CC, Doplnkové služby SS, Služba krátkych textových správ SMS a lokalizačné služby LCS. Služba CC zabezpečuje zriadenie, udržanie či ukončenie hovoru. SS zas podporuje presmerovanie hovorov, blokovanie hovoru či jeho podržanie, SMS umožňuje príjmanie a odosielanie textových správ protredníctvom SDCCH a SACCH kanálov. LCS iniciuje a podporuje zistenie polohy MS.

3 Projekt OSMOCOM

Každé mobilné zariadenie, ktoré je pripojené k celulárnej sieti pozostáva z procesoru, na ktorom beží closed-source firmware. Každý z takýchto firmwarov obsahuje chyby a mnoho z nich má bezpečnostný systém. Open-source projekty však poskytujú vyššiu úroveň bezpečnosti , nakoľko viacej ľudí môže skontrolovať daný kód a bezpečnostné chyby môžu byť opravené takmer okamžite. Veľa ľudí považuje pripojenie nezabezpečeného počítača do verejnej internetovej siete za veľmi nebezpečné. Pre ochranu svojich počítačov používajú rôzne firewally či antivírusy. Ale čo také mobilné zariadenia, ktoré sú neustále pripojené k verejnej sieti ? Vo väčšine prípadov neexistuje čistý spôsob ako možné chyby v takomto softwari opraviť, dokým výrobcovia týchto zariadení nevydajú jeho aktualizáciu, čo sa však stáva veľmi zriedkavo. Spoločnosti, ktoré majú prístup k týmto zdrojovým kódom firmwaru nemajú žiadny záujem o vylešenie tejto situácie. Preto riešením pre vylepšenie tohto problému je vytvorenie open-source projektu. A tak vznikol projekt Osmocom, ktorý umožňuje nahranie vlastného firmwaru do mobilného telefónu.

Myšlienkou projektu Osmocom je vytvorenie open-source GSM stacku, ktoré umožňuje užívateľovi vytvoriť a prevádzkovať mobilnú stanicu pomocou softwaru. Implementuje prvé tri vrstvy GSM protokolového zásobníku na strane klienta a taktiež ovládače pre zariadenie. Použitie tohto softwaru na kompatibilnom telefóne, umožňuje užívateľovi vytvárať a príjmať telefónne hovory, či posielať a príjmať textové správy. Súčasné mobilné zariadenia sa skladajú z dvoch separátnych častí a to počítača a modemu. Na počítači beží operačný systém Android, iOS alebo Symbian. Taký operačný systém však v skutočnosti môže v rámci GSM siete uskutočňovať pomerne málo funkcií. GSM modem je zložený z mikrokontroléru a DSP procesoru. DSP je čip určený pre algoritmy, ktoré sa využívajú pri spracovaní digitálnych signálov . Na mikrokotroléri aj na DSP procesore beží uzavretý firmware a v mnoho prípadoch je nepriateľský voči manipulácii s ním. To znamená, že spustenie nášho vlastného firmwaru na takomto zariadení by nebolo možné. Avšak sú aj telefóny, na ktorých by bolo možné spustiť vlastný zdrojový kód a to dokonca neničivým spôsobom. Na niektoré telefóny je možné nahrať vlastný firmware aj na DSP procesor a to vďaka RAM pamäti, keďže po vybratí batérie sa RAM vymaže a všetko v telefóne sa vráti do pôvodného stavu. V tejto kapitole bolo čerpané z nasledujúcich informačných zdrojov: [9], [10], [11], [12], [13], [14], [15].

3.1 OpenBSC

OpenBSC je software, ktorý spája GSM komponenty v projekte Osmocom. Nie je to len BSC, ktorú využíva štandartná GSM sieť, ale skladá sa z mnoho softwarových nástrojov a programov pre vytvorenie vlastnej malej GSM siete. OpenBSC plní funkciu, ktorú v tradičnej GSM sieti vykonávajú nasledujúce zložky: HLR, VLR, EIR, Auc, BSC a MSC. Prídavné funkcie EDGE a GPRS sú v projekte Osmocom realizované prostredníctvom programov OpenGGSN a OsmoSGSN. Pomocou

knižníc Libosmocore a Libosmo-Abis sú vytvorené dve hlavné OpenBSC aplikácie, Osmo-NITB a Osmo-BSC. Prostredníctvom Osmo-NITB a terminálových príkazov je možné ovládať a konfigurovať mnoho BTS staníc, bez naviazania spojenia so zložkami GSM siete ako je napríklad MSC. Osmo-BSC zas zabezpečuje pripojenie ku GSM IP sieti cez Abis rozhranie. Pre zriadenie IP spojenia cez A rozhranie, je nutná prítomnosť protokolu SCCP, čo zaisťuje knižnica Libosmo-SCCP.

3.2 OsmoBTS

OsmoBTS implementuje druhú a tretiu vrstvu GSM BTS a je použiteľný pre rôzne hardwarové L1 implementácie, kvôli chýbajúcemu GSM Um rozhraniu. V súčasnosti je OsmoBTS ešte stále vo fázi vývoja, hardwarovo podporuje zatiaľ iba sysmocom IP GSM BTS stanicu sysmoBTS. Projekt osmocom chce v budúcnosti umožniť podporu USRP, integráciou OpenBTS. V pláne je taktiež implementácia "virtual layer1", čo by umožňovalo spustenie BTS stanice bez použitia rádiovej vrstvy, prepojením OsmocomBB zásobníku cez TCP/IP pre sieťovú simuláciu a softwarové testovanie. Ako už bolo spomenuté, tento projekt je stále nedokončený a niektoré funkcie ešte stále chýbajú, ako napríklad handover. Rozšírením osmo-pcu o soketové rozhranie, bola umožnená funkcia GPRS.

3.3 Libosmocore/Libosmo-Abis

Libosmocore je knižnica rôznych potrebných funkcií, ktorú využívajú hlavne dva Osmocom projekty OpenBSC a OsmocomBB.

Libosmo-Abis obsahuje zdieľané GSM funkcie typické pre rozhranie A-bis a taktiež implementuje ovládače pre A-bis/IP a E1 rozhrania.

3.4 OsmoSGSN/OpenGGSN

OsmoSGSN, softwarová implementácia SGSN, zahŕňa funkcie GPRS Mobility Managment(GMM) a Session Managment(SM). Táto aplikácia spája OpenBSC cez Gb rozhranie a OpenGGSN skrz GTP protokol.

OpenGGSN plní funkciu GGSN v GPRS sieti a tvorí rozhranie medzi internetom a zvyškom mobilnej sieťovej infraštruktúry.

3.5 Osmo-PCU

V klasickej GSM sieti je súčasťou BSC jednotka paketového zriadenia PCU. Pomocou protokolov RLC a MAC zabezpečuje riadenie prenosových prostriedkov a konverziu paketov na rámce. Implementácie tejto jednotky sa v projekte Osmocom nazýva Osmo-PCU. Pre použitie Osmo-BTS a OpenBTS realizuje vlastné L1 rozhranie. Protokoly BSSGP a NS sú smerom k SGSN

realizované cez UDP/IP. Síce Osmo-PCU poskytuje užitočnú službu pre množstvo užívateľov, momentálne je však ešte nestabilná.

3.6 Komerčná BTS a OpenBSC

Jednou z možností prevádzky celulárnej siete je použitie OpenBSC spolu s komerčnou BTS. Problémom však je, že komerčne základňové stanice sú omnoho drahšie a ťažšie zakúpiteľné ako USRP. OpenBSC podporuje len zopár modelov a tieto jednotlivé základňové stanice sú ťažko zohnateľné pre obyčajných výskumníkov. Pokiaľ by užívateľ mal záujem o komerčnú BTS stanicu, tak najlepšou možnosťou je využiť ponuku od firmy Sysmocom GmbH, ktorá ponúka prednastavené GSM siete a s nimi spojené služby vrátane komerčných BTS staníc. Táto firma zároveň plánuje vydať nový typ SysmoBTS pre poskytnutie vhodnej alternatívy miesto komerčných BTS. Pripojením do výmenného L1 hardwaru je možné OpenBSC taktiež pripojiť k OsmoBTS.

4 Vývojový kit openBTS s USRP

4.1 Úvod do OpenBTS

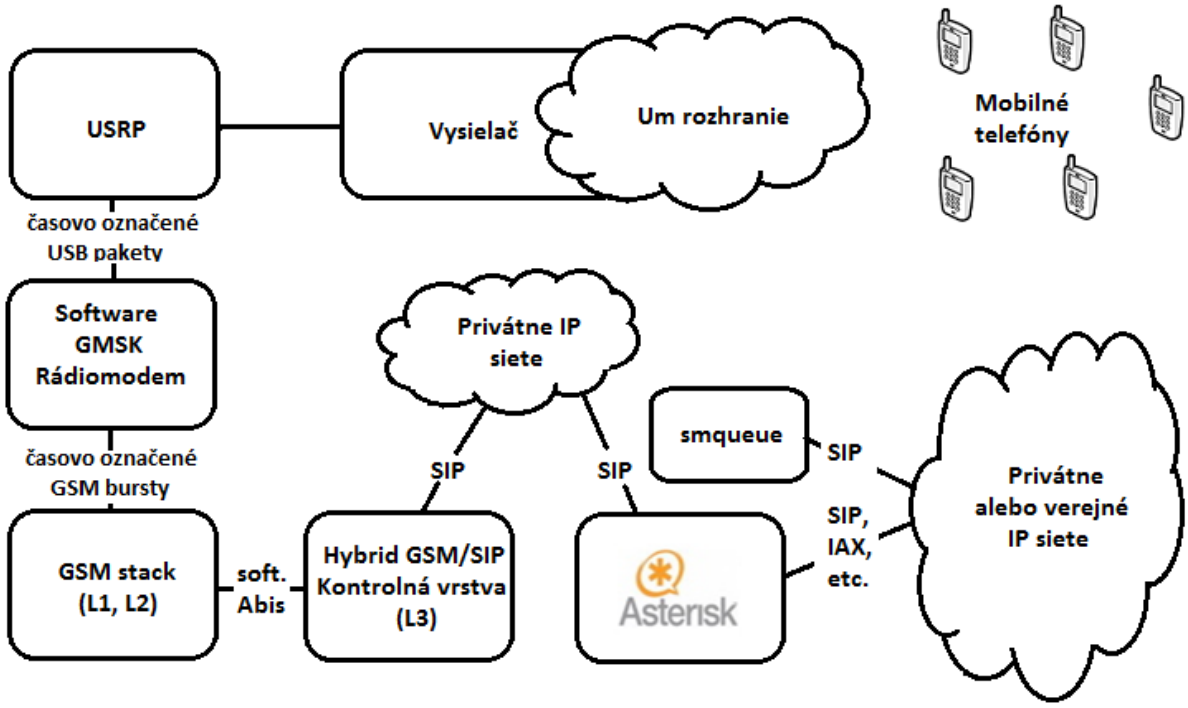
Vytvoriť vlastnú základňovú stanicu, ktorá by smerovala hovory po celom svete a navyše by bola prevádzkovaná za podstatne nižšie náklady ako súčasné technológie, dokáže asi málokto. Práve však open-source Unixová aplikácia s názvom OpenBTS poskytuje komukoľvek možnosť rozbehnúť mobilnú sieť za 1/10 prevádzkových nákladov, pretože systém je mnohokrát menší a spotrebuje omnoho menej energie ako súčasné technológie. Prostredníctvom tejto siete by si mohli vlastníci mobilných telefónov navzájom volať aj po celom svete, pokiaľ by bola sieť pripojená k internetu.

Projekt OpenBTS by mal zabezpečiť jednoduchší a lacnejší prístup k mobilným službám v chudobných oblastiach sveta so slabo vybudovanou infraštruktúrou, ale aj v oblastiach, ktoré sú ťažko dostupné ako sú napríklad ropné plošiny. V tejto kapitole som čerpal z nasledujúcich informačných zdrojov: [7], [8], [16], [17],[20],[22],[23].

4.2 Architektúra OpenBTS

Na rozdiel od súčasnej GSM, základňovej stanice BTS, ktorá potrebuje pre smerovanie prichádzajúcich a odchádzajúcich hovorov mobilnú spínaciu ustredňu MSC a je ovládaná pomocou riadiacej jednotky BSC, OpenBTS pre vytvorenie GSM siete využíva univerzálny softwarovo riadený vysielač USRP na L1 vrstve, ktorý predstavuje Um vzdušné rozhranie pre akýkoľvek mobilný telefón. Smerovanie a prepínanie hovorov je realizované softwarovou pobočkovou telefónnou ústredňou Asterisk VoIP na L3. Verejná verzia tohto projektu je vydávaná v zdrojovom kóde napísaného v jazyku C++ a implementuje prvé tri vrstvy štandartu GSM. Na obrázku 6 je graficky znázornená celková architektúra OpenBTS siete.

OpenBTS ktorý, tvorí rozhranie medzi Asteriskom a mobilnými telefónmi, namiesto spojenia s BSC riadiacou jednotkou, prečíta IMSI mobilného zariadenia a vytvorí obraz zariadenia SIP v sieti s ID, ktoré obsahuje načítanú identifikáciu. SIP zariadenie predstavujúce tento mobilný telefón sa pokúša registrovať s Asteriskom. Pokiaľ registrácia prebehla úspešne a tomuto zariadeniu bolo priradené telefónne číslo, mobilný telefón by mal byť dostupný vytočením tohto čísla a zároveň by mal byť schopný dovolať sa ďalším mobilným telefónom uložených rovnakým spôsobom v telefónnom zozname Asterisku.



Obrázok 6: *Architektúra OpenBTS*

4.2.1 Vrstvový model OpenBTS:

L0 - vysielač

L1 - funkcie FEC a TDM

L2 - LAPDm

L3 - funkcie pre správu rádiových zdrojov, GSM-SIP brána pre riadenie hovorov, GSM-SIP brána pre textové správy, GSM-SIP brána pre riadenie mobility

4.2.2 Hardwarové požiadavky

- Počítač**,** ktorý je potrebný pre spojenie s USRP doskou pomocou USB portu.
- USRP od spoločnosti Ettus Research, napríklad USRP1 či B100.
- Prídavné dosky, lepšiu kvalitu a pokrytie signálov.
- Anténa. Na jednu prídavnú dosku- jedna anténa.
- 52 Mhz hodiny. Pôvodná USRP doska je dodávaná s 64 Mhz kryštálovým oscilátorom a niektoré telefóny by neboli schopné prihlásenia do OpenBTS siete.
- Mobilné telefóny. Musia byť odomknuté pre akéhokoľvek operátora.
  SIM karty. Je možné použiť aj štandartnú SIM kartu alebo je možné si zakúpiť programovateľnú.

4.3 USRP

USRP vyvinutý spoločnosťou Ettus Research, je možné považovať za základný stavebný kameň celej OpenBTS architektúry, bez ktorého realizácia tohto projektu bola priam nemožná. Ako už bolo vyššie spomenuté, USRP je softwarovo riadený hardware, ktorý je pomerne lacný, môže byť ľahko prispôsobený vysielaču GSM a komunikuje s mobilnými telefónmi prostredníctvom vysielania a príjmania rádiových signálov. Na rozdiel od OpenBSC, však OpenBTS nie je možné pripojiť ku súčasnej GSM infraštruktúre. Možnosťou je pripojenie OpenBTS k OsmoBTS a cez OpenBSC do GSM siete. Systém USRP obsahuje základnú dosku, USB 2.0 rozhranie prostredníctvom, ktorého komunikuje s počítačom, programovateľné hriadlové pole (FPGA) a môže byť rozšíriteľné o ďalšie prídavné karty, pre vysielanie a príjem v rôznych frekvenčných pásmach. Napríklad pre frekvenčné pásmo 850 a 900 Mhz by sa použil model RFX 900 a pre pásmo 1800 a 1900 Mhz zas model RFX 1800.

Subsystémy základnej dosky USRP:

- A/D a D/A prevodník
- Rozhranie pre procesor
- Regulátor energie
- Generovanie a synchronizácia hodín
- FPGA

Obrázok 7: *USRP N210[22]*

4.4 USRP Hardware Driver UHD

Každý USRP je riadený prostredníctvom open-source UHD ovládača, ktorý je pracuje správne na skoro všetkých hlavných operačných systémoch, či už je to Linux alebo Windows. Tento ovládač je použiteľný jednak samostatne, ale aj s aplikáciami ako je napríklad Labview, GNU Radio, či už spomenuté OpenBTS. Ovládač UHD využívajú všetky zariadenia zo spoločnosti Ettus Research okrem USRP1, ktorý využíva GNU Rádio ovládač.

4.5 Smqueue

Pre zasielanie textových správ z jednej mobilnej stanice do inej, je v systéme OpenBTS využívaný server smqueue, ktorý plní funkciu prepínania správ. Jádro smqueue tvorí fronta správ čakajúcich na doručenie. Skrz množstvo pokusov o ich doručenie, správy čakajú v tejto fronte až do tej doby kým je správa potvrdená alebo označená za nedoručiteľnú. Smqueue rozoznáva dva druhy adries, a to konkrétne numerické adresy ISDN/E.164 a SIP uživateľské mená. Pomocou účastníckeho registru, smqueue prekladá všetky numerické adresy na SIP užívateľské mená. Za SIP užívateľské mená sú považované všetky tie adresy, ktoré nie sú zcela numerické. Pre realizáciu SMS, sa v OpenBTS nenachádza MSC a SMSC je nahradené spomínaným smqueue.

4.6 Subscriber registry

Subscriber registry je podobný ako konvenčný SIP register, avšak je rozšírený o funkcie pre podporu mobility a autentizácie. Tento register je v podstate databázový server s rozhraním pre spracovanie SIP REGISTER metód a poprípade SS7-MAP a DIAMETER rozhraní do PLMN, pre potrebu roamingu. Komunikácia OpenBTS jednotiek s SR prebieha prostredníctvom SIP. Ostatné sieťové prvky pristupujú k SR prostredníctvom SQL. Pre zredukovanie záťaže sú časti SR lokálne ukladané v BTS jednotkách.

4.7 Asterisk

Ďalšiou dôležitou súčasťou OpenBTS architektúry je voľne stiahnuteľná Unixová aplikácia Asterisk s vlastnosťami totožnými so štandartnou telefónnou ústredňou. Prostredníctvom protokolov IAX, H.323, SIP, MGCP a SCCP, umožňuje okrem VoIP , analógovej PSTN či ISDN digitálnej telefónie aj intertaktívny hlasový systém (IVR) a automatickú distribúciu volaní (ADC). IVR predstavuje hlasom alebo tónovou voľbou ovládaný terminál, ktorý plní funkciu operátora v zmysle prepojovania hovorov alebo zanechania hlasovej správy. ADC zas skupine terminálov dopravuje prichádzajúce hovory a zabezpečuje ich smerovanie. Na rozdiel do konvečnej GSM siete Asterisk eliminuje nutnosť použitia niektorých prvkov ako je HLR, MSC, atď.

V tejto bakalárskej práci budem prepájať dve Asterisk ústredne. Jednej ústredne, na ktorej beží OpenBTS a druhej, ktorú vytvorím na serveri Besip. Ústredne spojím najprv prostredníctvom SIP trunku a v ďalšej časti pomocou IAX trunku.

4.7.1 SIP protokol

Najznámejším VoIP protokolom je SIP, ktorý je zväčša používaný pre audio. SIP je peer-to-peer textovo orientovaný protokol, ktorý sa používa pre zostavenie, modifikáciu, ukončenie spojenia a pracuje na siedmej vrstve RM OSI. Cieľom tohto protokolu je zabezpečenie rovnakej funkcionality, ako je tomu v klasických PSTN sieťach. SIP ale poskytuje oveľa jednoduchšiu implementáciu nových služieb oproti PSTN a poniektoré spomedzi nich by mohli byť len ťažko zavedené v tradičných PSTN sieťach.

- **SIP komunikácia**

  Zložky SIP porotokolu medzi sebou komunikujú prostredníctvom následujúcich žiadostí:

  REGISTER – žiadosť o registráciu

  INVITE – žiadost o vytvorenie spojenia

  ACK – potvrdenie spojenia

  BYE – ukončenie spojenia

  CANCEL – žiadosť o ukončenie práve prebiehajúcej žiadosti INVITE

  OPTIONS – informácie o serveri

  Na SIP žiadosti je odpovedané následovnými číselnými odpoveďami:

  1XX – informačné správy (napr. 100 Trying)

  2XX – správa o úspešnom ukončení žiadosti (napr. 200 OK)

  3XX – správa o presmerovaní (napr. 305 Use Proxy)

  4XX – správa o chybe na strane klienta (napr. 403Forbidden)

  5XX – správa o chybe na serveri (napr. 500 Server Internal Error)

  6XX – správa o globálnom zlyhaní (napr. 606 NotAcceptable)

4.7.2 IAX protokol

Protokol IAX predstavuje Asterisk VoIP protokol. Vlastnosťou IAX protokolu je IAX trunking. Trunkovanie IAX spojenia je užitočné na sieťovej linke, ktorá by vykonávala viacero paralelných VoIP hovorov medzi dvomi systémami. Pomocou zapúdzrenia početných zvukových tokov do jedného paketu, IAX protokol zabezpečuje zníženie réžijných nákladov na datové spojenie a umožňuje ušetriť šírku pásma na vyťaženej sieťovej linke. Od SIP protokolu sa líši tým, že

signalizácia a média sú prenášané v pomocou jedného spojenia. IAX protokol popri minimalizácii počtu portov potrebných pre otvorenie na firewalle , skvele prechádza NATom.

- **IAX komunikácia**

  NEW – správa pre vytvorenie spojenia Úspešna žiadosť je potvrdená správou ACCEPT.

  ACCEPT – žiadosť pre vytvorenie spojenia sa potvdrzuje akonáhle správou ACCEPT, následne je vyslaná správa ACK a volajúci očakáva správu RINGING ,ANSWER, BUSY, PROCEEDING alebo HANGUP.

  REJECT – správa informuje o zlyhaní spojenia, čo mohlo byť sposobené zlým užívteľským menom, nesplenenou autentizáciou alebo zlým zadaným heslom.

  HANGUP – správa označujúca ukončenie spojenia.

  AUTHREQ - správa vyslaná pokiaľ je k zostaveniu spojenia vyžadovaná autentizácia.

  AUTHREP – správa zasiela potrebné autentizačné informácie vyžadované správou AUTHREQ.

5 Praktická realizácia GSM BTS s openBTS a scenáre použitia

5.1 Inštalácia OpenBTS

V prvom rade som si nainštaloval všetky požadované balíčky, ktoré sú potrebné pre správnu funkčnosť OpenBTS.

```
root@openbts:/# apt-get install autoconf libtool libosip2-dev
libortp-dev libusb-1.0-0-dev g++ sqlite3 libsqlite3-dev erlang
build-essential subversion python libboost-all-dev libusb-1.0-0-dev
python-cheetah doxygen python-docutils cmake unixodbc-dev ncursesdev
libsqliteodbc libxml2-dev
```

Ako už bolo spomenuté v predchádzajúcej kapitole, UHD zaisťuje komunikáciu medzi OpenBTS a USRP, preto je potreba tento ovládač nainštalovať.

```
bash -c 'echo "deb
http://files.ettus.com/binaries/uhd_stable/repo/uhd/ubuntu/`lsb_rele
ase-cs` `lsb_release -cs` main" >/etc/apt/sources.list.d/ettus.list'
apt-get update
apt-get install -t `lsb_release -cs` uhd
```

K inštalácii samotného OpenBTS som si vytvoril adresár */opt/OpenBTS/* a do tohto adresára som potom stiahol aktuálny zdrojový kód.

```
mkdir /opt/OpenBTS
cd /opt/OpenBTS
svn co http://wush.net/svn/range/software/public
```

Ku praktickej realizácii som využil USRP N210, ktorý požaduje podporu prevzorkovania. Z tohto dôvodu preto vykonáme následnú kompiláciu v adresári */opt/OpenBTS /public/openbts/trunk*.

```
cd /opt/OpenBTS/public/openbts/trunk
./autogen.sh
autoreconf –i
./configure --with-uhd --with-resamp
make
make install
```

Následne v zložke apps som vytvoril symbolický odkaz vysielača potrebný pre hardware USRP N210.

```
cd /opt/OpenBTS/public/openbts/trunk/apps
```

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

Konfigurácia OpenBTS sa nachádza v databázi sqlite, ktorá je umiestnená v adresári */etc/OpenBTS* a je generovaná prostredníctvom *.example.sql skriptov.

```
mkdir /etc/OpenBTS
cd /opt/OpenBTS/public/openbts/trunk
sqlite3 -init ./apps/OpenBTS.example.sql /etc/OpenBTS/OpenBTS.db
".quit"
```

Následne databázu je možné databázu editovať prostredníctvom nástroja sqlitebrowser

```
sqlitebrowser /etc/OpenBTS/OpenBTS.db
```

a upravíme v nej nasledujúce hodnoty:

```
GSM.Radio.Band 900
GSM.Radio.C0 1
Control.LUR.OpenRegistration .*
GSM.MNC 001
GSM.MCC 01
```

5.2 Inštalácia smqueue

Pre inštaláciu smqueue je nutné spustiť následovnú kompiláciu v zložke */opt/OpenBTS/public/smqueue/trunk*.

```
cd /opt/OpenBTS/public/smqueue/trunk
./autogen.sh
autoreconf –i
./configure
Make
```

Potom som si vytvoril databázu smqueue.db v adresári */etc/OpenBTS/*.

```
cd /opt/OpenBTS/public/smqueue/trunk/smqueue
sudo sqlite3 -init smqueue.example.sql /etc/OpenBTS/smqueue.db
".quit"
```

5.3 Inštalácia sipauthserve a Subscriber Registry

V prvom rade som si nastavil databázu Subscriber Registry.

```
sudo mkdir -p /var/lib/asterisk/sqlite3dir
cd /opt/OpenBTS/public/subscriberRegistry/trunk/configFiles/
sudo sqlite3 -init subscriberRegistryInit.sql
```

```
/var/lib/asterisk/sqlite3dir/sqlite3.db ".quit"
sudo mkdir /var/run/OpenBTS
```

Ďalej som nainštaloval sipauthserve, ktorý je potrebný pre SIP autentizačné služby. Po jeho inštalácii je ešte nutné vytvoriť databázu sipauthserve.db v zložke */etc/OpenBTS/*.

```
cd /opt/OpenBTS/subscriberRegistry/trunk
make
sqlite3 -init sipauthserve.example.sql /etc/OpenBTS/sipauthserve.db
".quit"
```

5.4 Inštalácia a konfigurácia Asterisku

Ako bolo spomenuté v predchádzajúcej kapitole, Asterisk je voľne stiahnuteľná unixová aplikácia s vlastnosťami totožnými ako klasická telefónna ústredňa, ktorá je využívaná pre smerovanie hovorov. Preto pre funkčnosť OpenBTS systeému je potrebná jeho inštalácia, ktorú vykonáme následovne:

```
sudo apt-get install asterisk libosip2-dev libortp7-*
```

Po úspešnom vykonaní inštalácie Asterisku je nutné upraviť niektoré jeho konfiguračné súbory. Najskôr som do súboru */etc/odbcinst.ini* pridal následovnú konfiguráciu.

```
[SQLite3]
Description=SQLite3 ODBC Driver
Driver=/usr/local/lib/libsqlite3odbc.so
Setup=/usr/local/lib/libsqlite3odbc.so
Threading=2
```

Ďalej v súbore /etc/odbc.ini som modifikoval sekciu asterisk.

```
[asterisk]
Description=SQLite3 database
Driver=SQLite3
Database=/var/lib/asterisk/sqlite3dir/sqlite3.db
# optional lock timeout in milliseconds
Timeout=2000
```

Následne som pridal symbolické odkazy pre súbory *odbc.ini* a *odbcinst.ini*.

```
root@openbts:/# cd /usr/local/etc
root@openbts:/# ln -s /etc/odbc.ini
ln -s /etc/odbcinst.ini
```

```
cd /root
ln -s /etc/odbc.ini .odbc.ini
ln -s /etc/odbcinst.ini .odbcinst.ini
```

V konfiguračnom súbore */etc/asterisk/modules.conf* je nutné povoliť automatické nahrávanie modulov.

```
autoload=yes
; noload => res_config_odbc.so
```

Aby Asterisk používal odbc databázu v real time móde, je potrebné upraviť konfiguračný súbor */etc/asterisk/extconfig.conf*.

```
[settings]
sipusers => odbc,asterisk,sip_buddies
sippeers => odbc,asterisk,sip_buddies
```

Do súboru *extensions.conf* som pridal kontext, ktorý umožňuje vyhľadávanie záznamov pre mobilné stanice pri volaní v databáze *sqlite3.db*.

```
[internal]
exten => _N.,1,Set(Name=${ODBC_SQL(select dial from dialdata_table
where exten = \"${EXTEN}\")})
exten => _N.,n,GotoIf($["${Name}" = ""] ?outbound-trunk,${EXTEN},1)
exten => _N.,n,Set(IPAddr=${ODBC_SQL(select ipaddr from sip_buddies
where name = \"${Name}\")})
exten => _N.,n,GotoIf($["${IPAddr}" = ""] ?outbound-
trunk,${EXTEN},1)
exten => _N.,n,Dial(SIP/${Name}@${IPAddr}:5062)
```

V súbore */etc/asterisk/res_odbc.conf* som upravil konfiguráciu podľa nasledujúcich riadkov:

```
[asterisk]
enabled => yes
dsn => asterisk
pre-connect => yes
```

V súbore */etc/asterisk/func_odbc.conf* som povolil použitie príkazov SQL v dialpláne.

```
[SQL]
dsn=asterisk
readsql=${ARG1}
```

Potom som nakopíroval konfiguračné súbory z adresára AsteriskConfig do zložky */etc/asterisk/*.

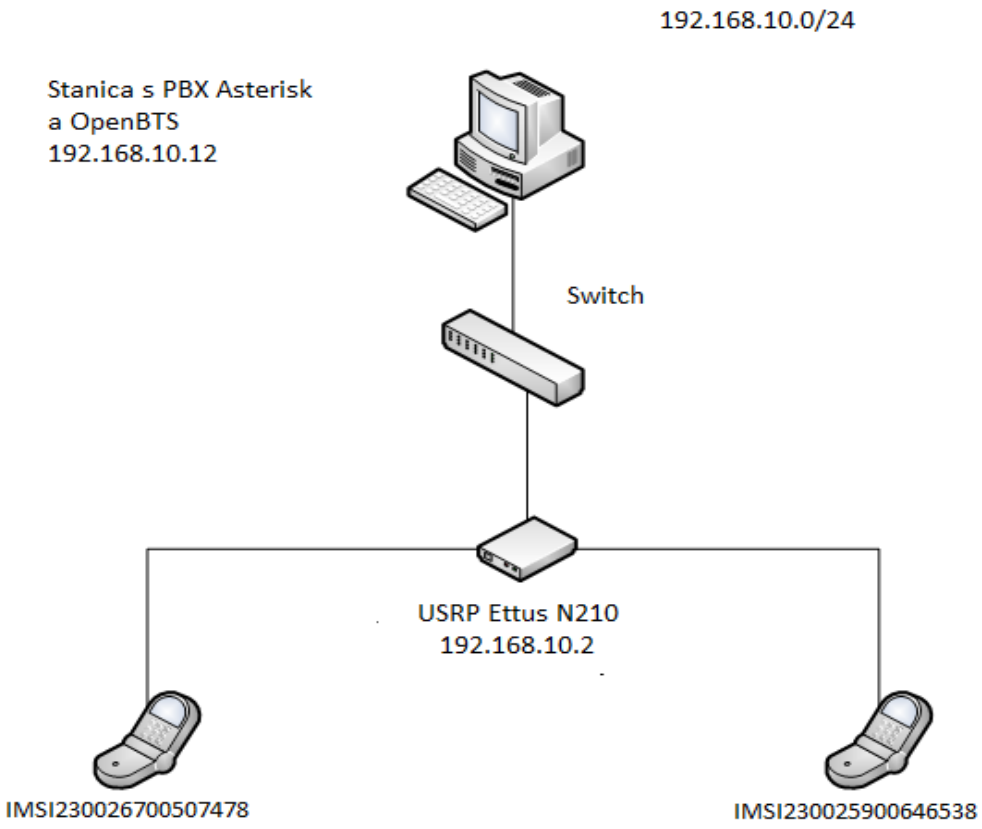
```
cp /opt/OpenBTS/openbts/trunk/AsteriskConfig/*.conf /etc/asterisk
```

OpenBTS loguje do súboru syslog a preto pre vytvorenie vlastného logovacieho súboru OpenBTS.log, som si vytvoril súbor */etc/rsyslog.d/OpenBTS.conf* s nasledujúcim obsahom:

```
local7.*/var/log/OpenBTS.conf
```

5.5 Spustenie

Akonáhle som nainštaloval jednotlivé prvky potrebné pre správnu funkčnosť OpenBTS, zostálo mi ešte prepojiť jednotlivé komponenty a registrovať účastníkov do konfiguračných súborov. Schému zapojenia pracoviska je možné vidieť na obrázku 8. Na počítači, ktorému som pridelil IP adresu 192.168.10.12 bežia paralelne programy smqueue, sipauthserve, OpenBTS a Asterisk. Pomocou ethernet káblu je prepojený počítač so switchom a ten následne s USRP N210, ktorý má pridelenú IP adresu 192.168.10.2. Dostupnosť USRP je možné overiť prostredníctvom príkazu ping.



Obrázok 8: *Schéma zapojenia*

Po vykonaní všetkých predchádzajúcich krokov následne ešte potrebujem overiť kontrolu funkčnosti pomocou príkazov *uhd_find_devices* a *uhd_usrp_probe*. Pokiaľ dojde pri kontrole k chybám ako je vidno na obrázku 9, je nevyhnutné aktualizovať firmware a FPGA hardwaru USRP N210.

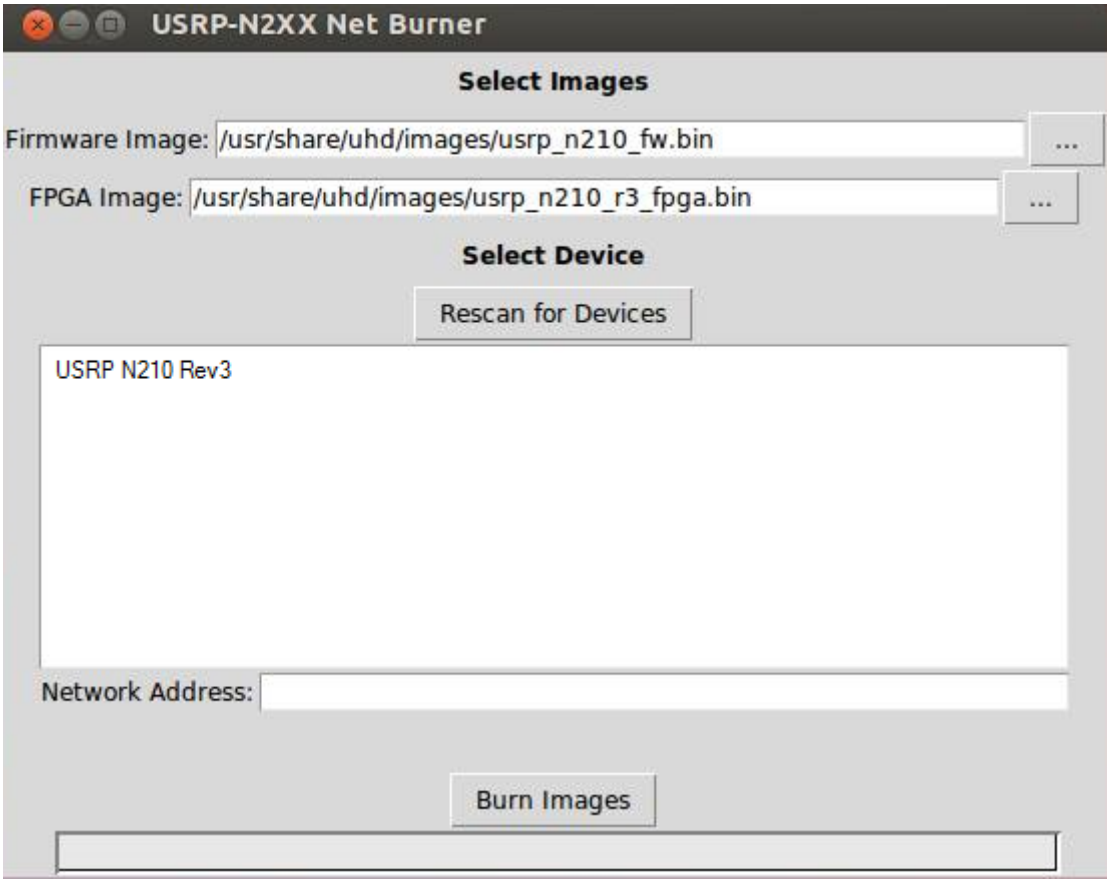
```
root@openbts:/opt/OpenBTS/public/openbts/trunk# uhd_find_devices
linux; GNU C++ version 4.6.3; Boost_104601; UHD_003.005.004-release

No UHD Devices Found
root@openbts:/opt/OpenBTS/public/openbts/trunk# uhd_usrp_probe
linux; GNU C++ version 4.6.3; Boost_104601; UHD_003.005.004-release

Error: LookupError: KeyError: No devices found for ----->
Empty Device Address
```

Obrázok 9: *No UHD Devices Found*

V adresári */usr/share/uhd/utils* prostredníctvom terminálu som spustil príkazom */usr/share/uhd/utils/usrp_n2xx_net_burner_gui.py* grafický nástroj pre aktualizáciu USRP:



Obrázok 10: *GUI pre aktualizáciu USRP*

Do položky Firmware Image nahráme zo zložky */usr/share/uhd/images* súbor *usrp_n210_fw.bin* a do FPGA Image z tej istej zložky vybereme súbor *usrp_n210_r3_fpga.bin* a proces aktualizácie spustíme kliknutím na tlačítko Burn Images. Následne by už zariadenie malo fungovať správne.

Teraz už nič nebráni spusteniu prevádzky vlastnej GSM siete už je nutné len paralelné spustenie smqueue, sipauthserve, OpenBTS a Asterisku. Ku každej spustenej aplikácii pridávam výpis so správnym výstupom v termináli.

- **Spustenie smqueue**

```
cd /opt/OpenBTS/public/smqueue/trunk/smqueue
sudo ./smqueue
ALERT 140084989957952 smqueue.cpp:2421:main: smqueue (re)starting
smqueue logs to syslogd facility LOCAL7, so there's not much to see
here.
```

- **Spustenie sipauthserve**

```
cd /opt/OpenBTS/public/subscriberRegistry/trunk
./sipauthserve
ALERT 140615443003200 sipauthserve.cpp:214:main:./sipautserve
(re)starting
```

- **Spustenie OpenBTS**

```
cd /opt/OpenBTS/public/openbts/trunk/apps
./OpenBTS
System ready
Use the OpenBTSCLI utility to access CLI
```

- **Spustenie Asterisku**

```
asterisk –vvvvvr
==========================================================================
Connected to Asterisk 11.6.0 currently running on Besip (pid = 6260)
Besip*CLI>
```

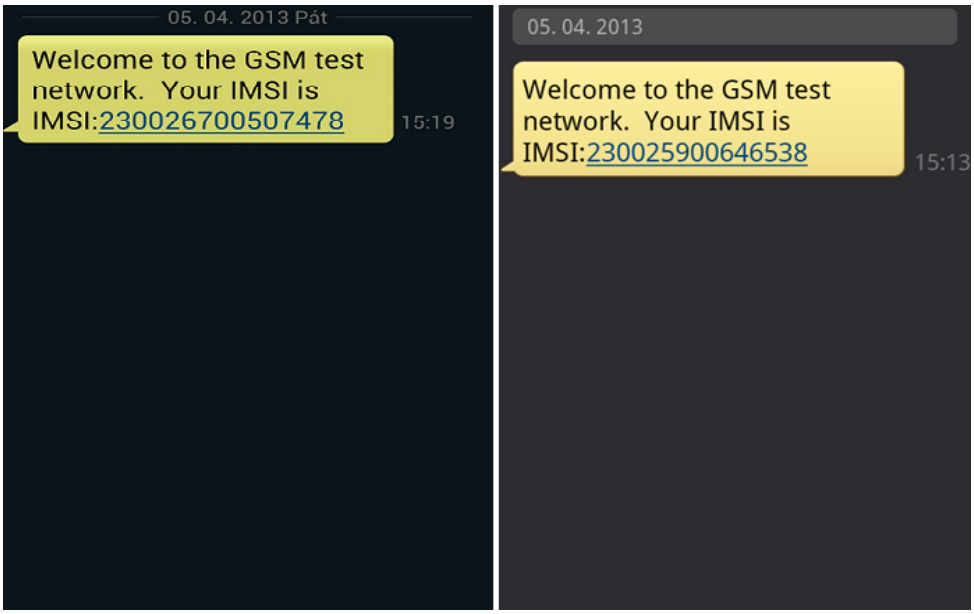
5.6 Scenáre použitia

Existuje množstvo VoIP protokolov, ktoré môžeme využiť pre prepojenie Asterisku s privatnými alebo verejnými IP sieťami. Asterisk podporuje protokoly SIP, IAX, H.323, MGCP, SCCP a VoFR.

V tejto bakalárskej práci sa však budeme venovať dvom najhlavnejším protokolom: Session Initiation Protocol (SIP) a InterAsterisk eXchange Protocol (IAX). Vytvorím si vlastného SIP/IAX trunk providera na serveri BESIP, čím rozšírim sieť OpenBTS o možnosť prakticky sa dovolať kdekoľvek prostredníctvom internetu.

5.6.1 Konfigurácia spojenia mobilných staníc v sieti OpenBTS

Na obrázku 11 je možné vidieť, že v momente ako som spustil OpenBTS a pripojil mobilné zariadenia k sieti, obdržal som uvitačné sms správy, ktoré obsahovali ich IMSI čísla.



Obrázok 11: *Uvítacia správa*

Tieto IMSI čísla je potrebné následne vložiť do konfiguračných súborov Asterisku a zároveň do SQL databáze. Buď je možnosť jednotlivé konfigurácie vložiť manuálne alebo pomocou skriptu alebo mnou vytvorený C++ program, ktorý po zadaní vstupného parametru (IMSI čísla), upraví konfiguráčné súbory a SQL databázu Asterisku aby bola umožnená komunikácia medzi mobilnými stanicami v sieti. Po nainštalovaní Subscriber registry sa v jeho adresári */opt/OpenBTS/public/subscriberRegistry/trunk/sqlite3/* nachádza hlavičkový súbor *sqlite3.h* a *sqlite3.c*. Prostredníctvom týchto dvoch súborov som si vytvoril objektový súbor *sqlite3.o*, ktorý využíva program *astprg.cpp* pre svoju správnu funkčnosť.

V prvom rade sa prostredníctvom terminálu treba dostať do zložky, kde sa tento program nachádza a pomocou príkazu make ho skompilujeme. Následne už je možné vkladať IMSI čísla do jednotlivých konfigurácii príkazom *./astprg <IMSI>*.

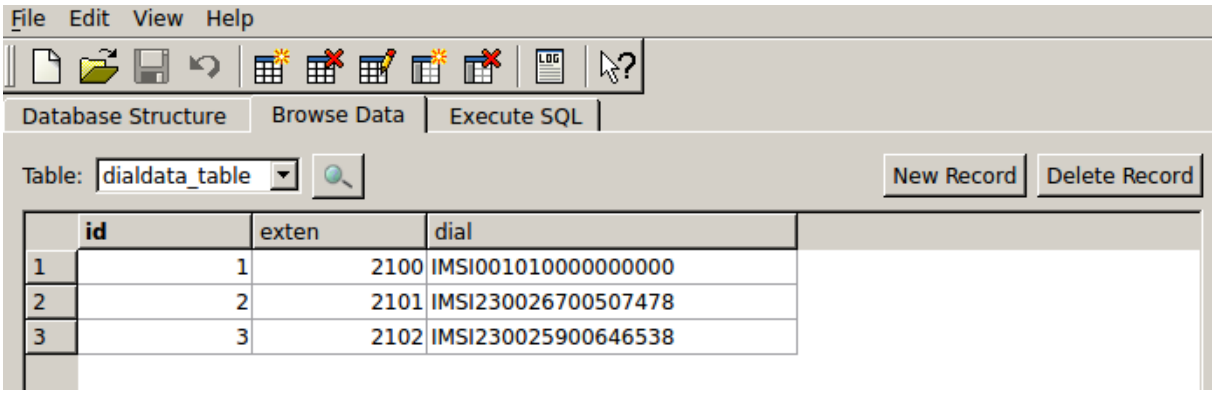
```
root@openbts:/home/openbts/Desktop/dist# ./astprg IMSI230025900646538
Opening database...OK
IMSI not used, creating new record.
Inserting to db: id(2), number(2101), imsi(IMSI230025900646538)
Inserting to dialdata_table...OK
Inserting to SIP_BUDDIES...OK
/etc/asterisk/sip.conf updated.
/etc/asterisk/extensions.conf updated.
Closing DB...OK
root@openbts:/home/openbts/Desktop/dist# ./astprg IMSI231060900692149
Opening database...OK
IMSI not used, creating new record.
Inserting to db: id(3), number(2102), imsi(IMSI231060900692149)
Inserting to dialdata_table...OK
Inserting to SIP_BUDDIES...OK
/etc/asterisk/sip.conf updated.
/etc/asterisk/extensions.conf updated.
Closing DB...OK
```

Obrázok 12: *Registrácia IMSI*

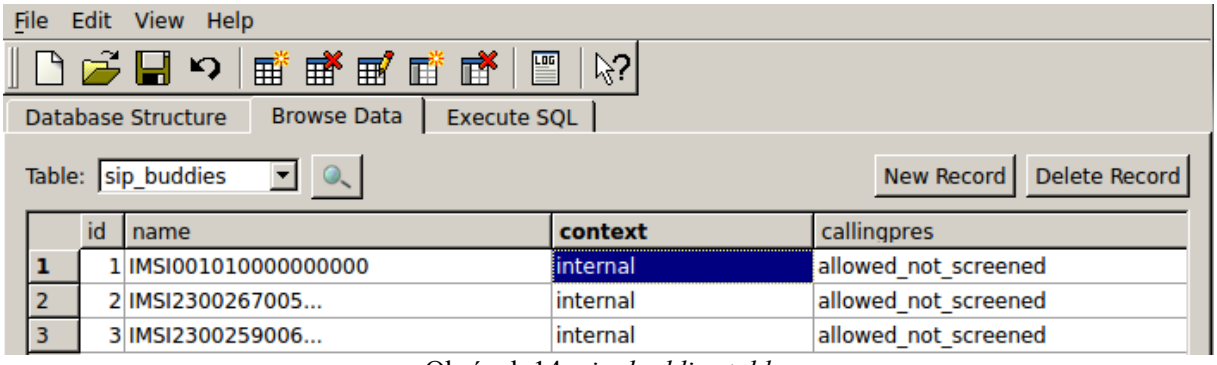
Na obrázku 12 je možné vidieť, že program astprg v prvom rade pomocou funkcie *searchForIMSI()* skontroluje údaje v databáze a zistí či sa náhodou dané IMSI číslo už v databáze nenachádza. V prípade ak sa dané IMSI číslo už v databáze nachádza, tak je ignorované. Pokiaľ v databáze nie je, tak program prostredníctvom funkcie *generateID()* mu vygeneruje ID číslo tak, že skontroluje najväčšie ID v databáze, k nemu pripočíta 1. Na podobnom princípe funguje aj funkcia *generatePhoneNumber()*, ktorá vytvorí telefónne číslo pre vložené IMSI. Následne funkciami *insertRecordToDialdata() a insertRecordToSipBuddies()* vloží IMSI číslo spolu s vygenerovanými číslami do dialdata tabuľky (obr. 13) a sip_buddies tabuľky (obr. 14). Potom už je potrebné len vložiť IMSI do konfiguračných súborov Asterisku, čo zabezpečujú funkcie *createRecordInSipConf()* a *createRecordInExtensionsConf()*.

```
bool imsiFound = searchForImsi(db, imsi);
if (!imsiFound)
{
std::cout << "IMSI not used, creating new record." << std::endl;
      int id = generateId(db);
      std::string number = generatePhoneNumber(db);
insertRecordToDB(db, id, number, imsi);
      createRecordInSipConf(number, imsi);
```

```
        createRecordInExtensionsConf(number, imsi);
}
else
{
        std::cout << "IMSI is used, ignoring." << std::endl;
}
        sqlite3_close(db);
        return EXIT_SUCCESS;
}
```



Obrázok 13: *dialdata_table*



Obrázok 14: *sip_buddies table*

Podľa nasledujúceho vzoru program astprg vytvorí sekciu s IMSI číslom mobilnej stanice v súbore */etc/asterisk/sip.conf*.

Vzor:

```
/*
[IMSI230025900646538]
callerid=2102
type=friend
host=dynamic
```

```
canreinvite=no
context=remote
allow=gsm
dtmfmode=RFC2833
*/
void createRecordInSipConf(std::string number, std::string imsi)
{
      std::ofstream outfile;
      outfile.open(sipConfPath.c_str(), std::ios_base::app);
      outfile
      << "[" << imsi << "]" << std::endl
      << "callerid=" << number << std::endl
      << "canreinvite=no" << std::endl
      << "type=friend" << std::endl
      << "context=remote" << std::endl
      << "allow=gsm" << std::endl
      << "host=dynamic" << std::endl
      << "dtmfmode=RFC2833" << std::endl
      << std::endl;
      outfile.close();
}
```

Sekcia IMSI230025900646538 bude dostupná pod číslom 2102 a obsiahnuté v nej budú nasledujúce atributy:

- callerid = 2102 pridelené telefónne číslo
- canreinvite = no akonáhle je zriadený telefónny hovor, posielanie žiadosti REINVITE je zastavené
- type = friend umožňuje prichadzajúce aj odchadzajúce hovory
- context = remote definícia kontextu uloženého v extensions.conf, podľa ktorého bude mobilná stanica realizovať odchádzajúce hovory
- host = dynamic zariadenie sa môže prihlásiť k ústredne z akejkoľvek IP adresy
- allow = gsm povolenie GSM kodeku
- dtmfmode = RFC2833 nastavenie signalizácie

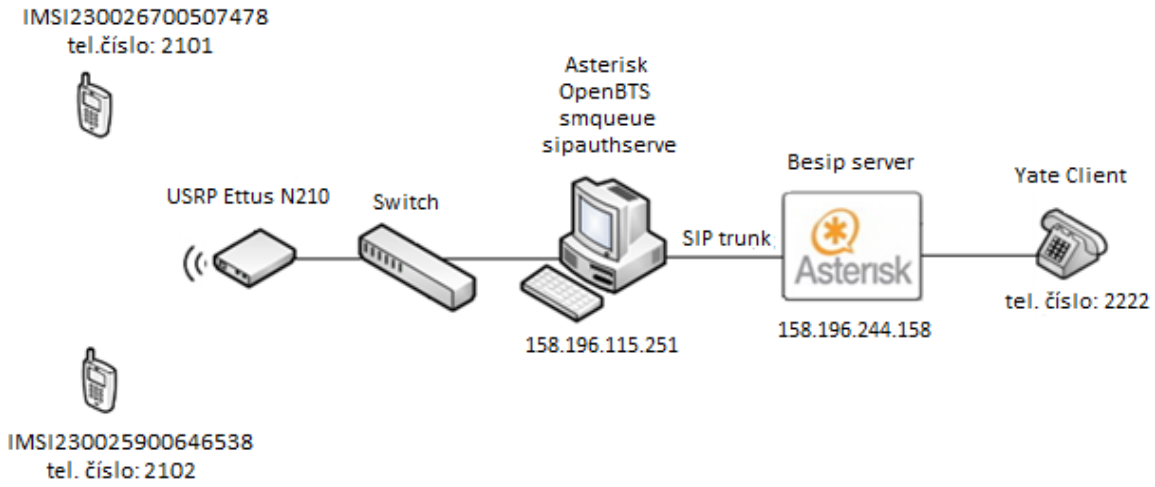
Aby bolo možné zostaviť telefónny hovor je potrebné do kontextu *[remote]* v súbore */etc/asterisk/extensions.conf* vložiť konfiguráciu s možnosťou vytočenia inej mobilnej stanice.

Vloženie tejto konfigurácie je zabezpečené funkciou *createRecordInExtensionsConf()* v programe astprg. Premenná number predstavuje vygenerované telefónne číslo a do premennej imsi je vložené IMSI číslo mobilnej stanice vložené pri spúšťaní programu.

```
// remote sekcia nájdená, vkladanie nového záznamu
if (line.compare(extensionsConfSection) == 0)
{
Outfile << "exten => " << number << ",1,Dial(SIP/" << imsi <<
@127.0.0.1:5062)" << std::endl;
}
[remote]
exten => 2101,1,Dial(SIP/IMSI230026700507478@127.0.0.1:5062)
exten => 2102,1,Dial(SIP/IMSI230025900646538@127.0.0.1:5062)
```

5.6.2 Pripojenie na SIP trunk providera

V tomto scenári priblížim ako si vytvoriť vlastného SIP providera a ako k nemu pripojiť môj Asterisk systém. Schému zapojenia pre tento scenár je možné vidieť na obrázku 15.



Obrázok 15: *Pripojenie na SIP trunk providera*

Najskôr som nakonfiguroval SIP trunk na počítači s IP adresou 158.196.115.251. Do konfiguračného súboru *sip.conf* bolo potrebné do sekcie *[general]* vložiť riadok pre registráciu, ktorý pozostáva z uživateľského mena *siptr*, predstavujúce SIP trunk, ktorý bude definovaný na strane serveru. Ďalej nasleduje heslo trunk, IP adresa serveru a port, cez ktorý má komunikácia prebiehať.

Následne už bolo potrebné len nakonfigurovať samotný trunk. Obojsmernú prevádzku, čiže umožnenie prichádzajúcich aj odchádzajúcich hovorov som zabezpečil príkazom *type=friend*. Trunk

siptr1 sa môže pripojiť k ústredne z akéjkoľvek IP adresy, čo umožňuje príkaz *host=dynamic*. Riadkom *trunk=yes* definujem, že sa bude jednať o SIP trunk. Do príkazu *username* bolo potrebné pridať uživateľské meno trunku na strane serveru. Aby Asterisk monitoroval SIP trunk, či je stále online som zabezpečil pridaním konfigurácie *qualify=yes* a na záver riadkom *context=internal* odkazujem na kontext v súbore *extensions.conf*, kde sa nachádzajú pravidlá, podľa ktorých sa bude `siptr` riadiť pri realizácii telefónnych hovorov.

```
[general]
register => siptr:trunk@158.196.244.158:5060/siptr
[siptr1]
type=friend
host=dynamic
trunk=yes
secret=trunk
username=siptr
qualify=yes
context=internal
```

Na strane serveru Besip bolo taktiež nutné zadefinovať trunk. Konfigurácia je takmer totožná s predošlou konfiguráciou, na strane počítača s OpenBTS. Rozdiel je v registračnom riadku, kde odkazujem na trunk definovaný na strane počítača s OpenBTS, ktorý beží na IP adrese 158.196.244.158. Ako som spomínal na serveri bude nakonfigurovaný trunk *siptr*, ktorého parametre budú takmer rovnaké ako v predchádzajúcom prípade. Jedine čo je treba pozmeniť, tak je položka *username,* do ktorej treba vložiť uživateľské meno trunku na opačnej strane.

```
[general]
...
register=> siptr1:trunk@158.196.115.251:5060/siptr1
...
[siptr]
...
username=siptr1
...
```

Zároveň som zadefinoval telefónneho účastníka s telefónnym číslom 2222, ktorý bude predstavovať softphone pripojený k serveru Besip, na ktorý budem telefonovať z môjho mobilného zariadenia pripojenému k sieti OpenBTS .

```
[2222]
callerid=2222
type=friend
qualify=yes
host=dynamic
context=remote
username=2222
```

Po upravení súboru sip.conf na obidvoch stranách, som potreboval upraviť ešte dialplán Asterisku. Do súboru extensions.conf bolo treba pridať pravidlá, ktoré mi umožňujú dovolať sa na číslo 2222 prostredíctvom trunku siptr1, ktoré predstavuje softphone pripojený k serveru Besip. V prípade, že mobilná stanica vytočí číslo 2222, vykonajú sa pravidlá z kontextu *[remote]*. Pravidlom *Dial()* sa pokúsi o naviazanie spojenia, následuje pravidlo *Congestion()*, ktoré by v prípade nedosiahnuteľnosti volaného zariadenia spustilo signalizáciu o obsadení a na záver *Hangup()* predstavuje ukončenie hovoru.

```
[remote]
exten => 2222,1,Dial(SIP/siptr1/${EXTEN})
exten => 2222,n,Congestion()
exten => 2222,n,Hangup()
```

Podobne som upravil dialplan aj na strane serveru, aby bolo možné uskutočniť telefónny hovor medzi softphonom a mobilnou stanicou s číslom 2102 v sieti OpenBTS. V serverom konfiguračnom súbore *extensions.conf* som musel ešte pridať kontext *[internal]*, ktorý využíva SIP trunk pre prichádzajúce hovory.

```
[internal]
exten => 2222,1,NoOp()
exten => 2222,n,Dial(SIP/${EXTEN},30)
exten => 2222,n,Hangup()
[remote]
exten => 2102,1,Dial(SIP/siptr/${EXTEN})
exten => 2102,n,Congestion()
exten => 2102,n,Hangup()
```

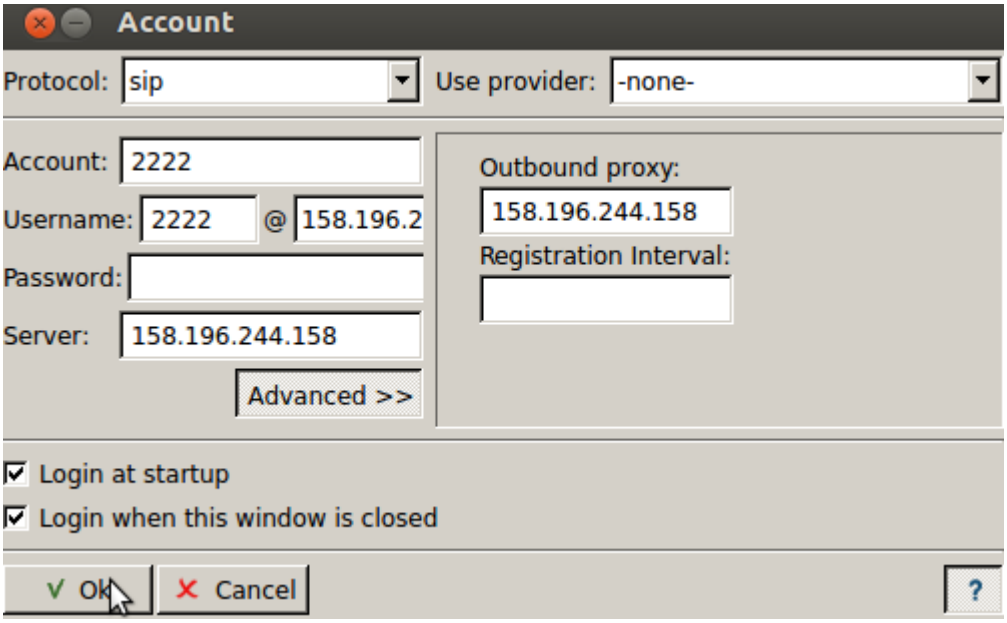
Na záver mi už chýba len nakonfigurovať softphone, prostredníctvom ktorého budem volať na mobilné stanice v sieti OpenBTS a samozrejme aj hovory príjmať. Pre tento scenár som si vybral softphone Yate Client, ktorý nainštalujem nasledujúcimi príkazmi v termináli.

```
root@openbts:/# svn checkout http://voip.null.ro/svn/yate/trunk yate
root@openbts:/# cd yate
root@openbts:/#./autogen.sh
root@openbts:/#./configure
root@openbts:/# make
root@openbts:/# make install
```

Následne už môžem spustiť softphone Yate v termináli príkazom:

```
root@openbts:/# yate
```

Po úspešnom vykonaní všetkých predchádzajúcich krokov, som už len potreboval zaregistrovať softphone s telefónnym číslom 2222 na Asterisk server. Obrázok 16 popisuje ako takýto účet nastaviť.



Obrázok 16: *Konfigurácia Yate*

5.6.3 Zabezpečenie SIP trunku prostredníctvom SIP TLS

V prvom rade je pre zabezpečenie SIP protokolu potrebné nainštalovať open source SSL/TLS knižnice – openssl.

```
root@openbts:/# apt-get install openssl
```

Následne som vytvoril zložku keys v adresári /etc/asterisk/, kde budem generovať potrebné certifikáty. Najskôr som si teda vygeneroval 1024 bitový kľúč.

```
root@openbts:/# mkdir /etc/asterisk/keys
root@openbts:/etc/asterisk/keys# openssl genrsa -des3 -out ca.key
1024
```

Pokračoval som vygenerovaním cerifikátu request.pem a následne som si certifikát aj sám podpísal.

```
root@openbts:/etc/asterisk/keys# openssl req -new -key key.pem -out
request.pem
root@openbts:/etc/asterisk/keys# openssl x509 -req -days 3650 -in
request.pem -signkey key.pem -out certificate.pem
```

Nakoniec som spojil vygenerovaný kľúč key.pem a certifikát certificate.pem do jedného súboru siptr1.pem.

```
root@openbts:/etc/asterisk/keys# cp certificate.pem siptr1.pem
root@openbts:/etc/asterisk/keys# cat key.pem >> siptr1.pem
```

V konfiguračnom súbore `sip.conf` som povolil TLS šifrovanie, definoval cestu k mojemu súboru s kľúčom a certifikátom serveru (siptr1.pem) a povolil iba šifrovaný prenos pre komunikáciu prostredníctvom SIP trunku.

```
[global]
tlsenable=yes                ;povolenie TLS
tlsbindaddr=0.0.0.0          ;pripojenie TLS k IP
tlscertfile=/etc/asterisk/keys/siptr1.pem  ;cesta k certifikátu
tlsdontverifyserver=yes      ;overenie či je Asterisk klient
tlscipher=DES-CBC3-SHA       ;šifrovací algoritmus
tlsclientmethod=tlsv1        ; verzia TLS/SSL
register=>tls://siptr1:trunk@158.196.115.251:5061/siptr
[siprt1]
transport=tls                ;povolenie TLS prenosu
```

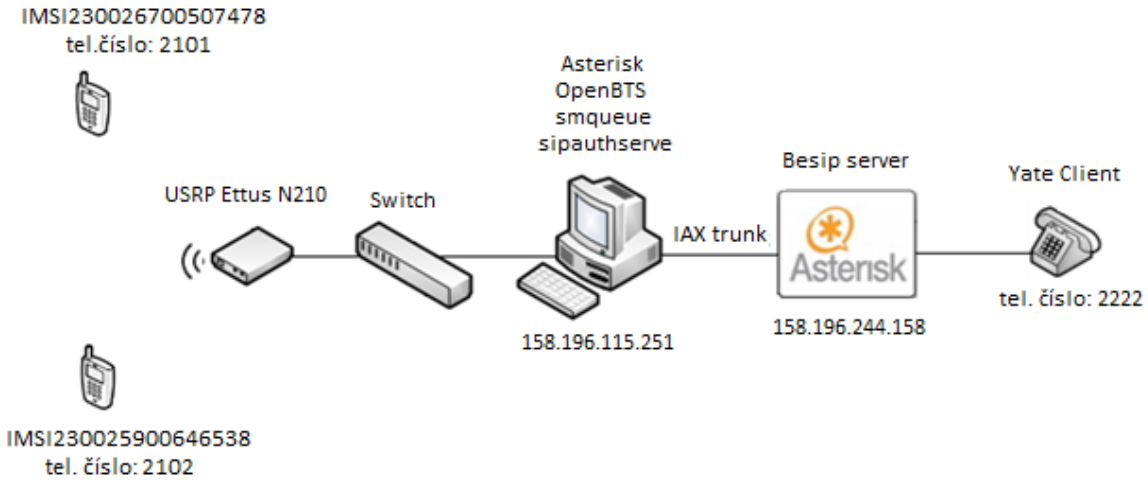
Po nakonfigurovaní SIP TLS aj na strane serveru a zinicializovaní spojenia medzi obidvoma ústredňami, si medzi sebou vymenia patričné šifrovacie informácie a následne už medzi nimi dochádza len k šifrovanému prenosu aplikačných dát. Celý tento proces je možné vidieť na obrázku 17.

158.196.244.158	158.196.194.131	TLSv1	132	Client Hello
158.196.194.131	158.196.244.158	TCP	68	sip-tls > 49416 [ACK] Seq=1 Ack=65 Win=27264 Len=0 TSval=814981 TSecr=34058604
158.196.194.131	158.196.244.158	TLSv1	754	Server Hello, Certificate, Server Hello Done
158.196.244.158	158.196.194.131	TCP	68	49416 > sip-tls [ACK] Seq=65 Ack=687 Win=17536 Len=0 TSval=34058607 TSecr=814981
158.196.244.158	158.196.194.131	TLSv1	258	Client Key Exchange, Change Cipher Spec, Encrypted Handshake Message
158.196.194.131	158.196.244.158	TLSv1	294	Encrypted Handshake Message, Change Cipher Spec, Encrypted Handshake Message
158.196.244.158	158.196.194.131	TLSv1	702	Application Data, Application Data
158.196.194.131	158.196.244.158	TLSv1	662	Application Data, Application Data
158.196.244.158	158.196.194.131	TLSv1	694	Application Data, Application Data
158.196.194.131	158.196.244.158	TLSv1	662	Application Data, Application Data
158.196.244.158	158.196.194.131	TLSv1	686	Application Data, Application Data
158.196.194.131	158.196.244.158	TLSv1	694	Application Data, Application Data

Obrázok 17: *SIP TLS (Wireshark)*

5.6.4 Pripojenie na IAX trunk providera

Na obrázku 18 je ukázané schéma zapojenia pre tento scenár. Opäť budem používať softphone Yate Client s číslom 2222. Rozdiel oproti predchádzajúcej konfigurácii je v použitom trunku, keďže sieť OpenBTS a Asterisk bežiaci na Besip serveri budú spojené pre zmenu prostredníctvom IAX trunku.



Obrázok 18: *Pripojenie na IAX trunk providera*

V tomto prípade sa však nebude praktizovať nastavenie v súbore sip.conf, ale v súbore iax.conf, ktorý sa taktiež nachádza v hlavnom adresári aplikácie Asterisk. Nastavenie IAX trunku je takmer totožné s konfiguráciou SIP trunku. Pre tento scenár sa bude využívať port 4569, ktorý využíva IAX pre svoju signalizáciu. Ďalšiou zmenou je pridanie riadku *autokill=yes*, ktorý zabráňuje preťaženiu v prípade, že je host nedostupný do určitého časového intervalu. Každý pokus o zincializovanie hovorového spojenia musí byť potvrdený do 2 sekúnd, inak bude zrušené. Možnosťou *deny* blokujem všetky IP adresy aby sa nemohli autentizovať a explicitne povolujem autentizáciu pre IP adresu serveru príkazom *permit*. Šifrovanie v IAX je zabezpečené prostredníctvom *encryption=yes*.

```
[general]
...
autokill=yes
port=4569
register=>iaxtr:trunk@158.196.244.158
encryption=yes        ;šifrovanie IAX
...
[iaxtr1]
auth=md5              ;md5 autentizácia
type=friend           ;prichádzajúce aj odchádzajúce hovory
host=dynamic          ;môže sa pripojiť z akejkoľvek IP
trunk=yes             ;nastavenie módu pre IAX trunk
secret=trunk          ;heslo pre autentizáciu
username=iaxtr        ;uživateľské meno trunku na opačnej strane
qualify=yes           ;monitorovanie či je host aktívny
context=internal      ;kontext pre prichádzajúce hovory
disallow=all          ;zakáz všetkých kodekov
allow=alaw            ;povolenie alaw kodeku pre EU
allow=gsm             ;povolenie GSM kodeku
deny=0.0.0.0/0.0.0.0 ;zákaz autentizácie všetkým adresám
permit=158.196.244.158/255.255.255.255 ;povolenie autentizácie
```

Podobne som definoval trunk aj na strane serveru, konfigurácia je rozdielná len v registračnom riadku, kde je potrebné zadefinovať IP adresu počítača, ku ktorému je pripojené USRP a zadefinovať SIP trunk *iaxtr*.

```
[general]
...
register=>iaxtr1:trunk@158.196.195.187
...
[iaxtr]
...
username=iaxtr1
permit=158.196.115.251/255.255.255.255
...
```

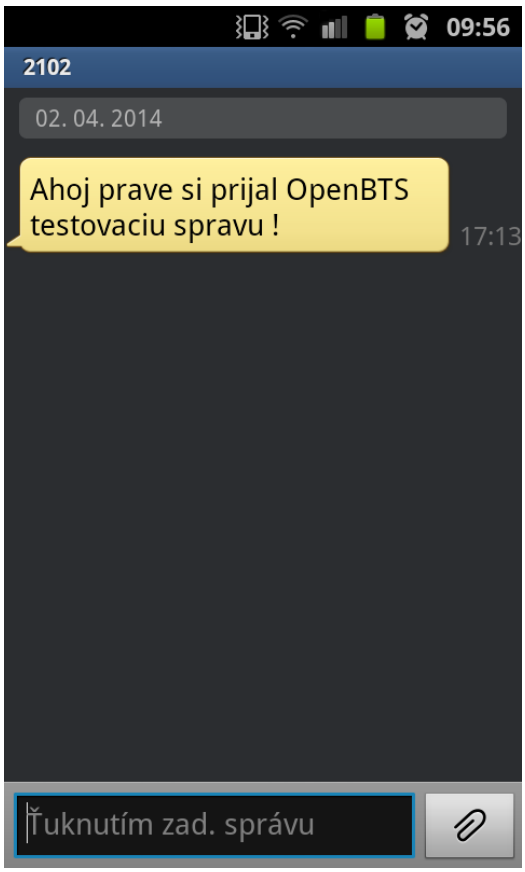
Ďalej som už len upravil pravidlá v súbore *extensions.conf* pre umožnenie telefónnych hovorov prostredníctvom IAX trunku. Či už na strane serveru alebo siete OpenBTS, v kontexte *[internal]* nebola potrebná žiadna zmena. Úpravy som urobil len v kontexte *[remote]*, kde v pravidle *Dial()* som nahradil SIP za IAX2, aby sa pri volaní na číslo 2102 (resp. 2222), použil IAX trunk, ktorý som nakonfiguroval v súbore *iax.conf*.

```
[remote]
...
;na strane serveru
exten => 2102,n,Dial(IAX2/iaxtr/${EXTEN})
;na strane OpenBTS
exten => 2102,n,Dial(IAX2/iaxtr1/${EXTEN})
...
```

6 Zhodnotenie dosiahnutých výsledkov

6.1 Realizácia hovoru a posielania správ v sieti OpenBTS

Pre tento scenár som použil dva mobilné telefóny Samsung Galaxy Ace2 a Nokiu 6700. Pri realizácii sa nevyskytol žiadny problém s posielaním sms správ alebo uskutočňovaním telefónnych hovorov medzi všetkými mobilnými stanicami v mojej vlastnej sotwarovo riešenej mobilnej siete, čo dosvedčujú, jednak obrázok 19 z mobilného telefónu a taktiež obrázok 20, ktorý zachytáva komunikáciu medzi mobilnými zariadeniami prostredníctvom aplikácie Wireshark, ktorej hlavnou úlohou je zachytávanie komunikácie v počítačových sieťach. Na tomto obrázku môžme vidieť, že telefónne číslo 2101 s IMSI230025900646538 je vyzvané metódou INVITE ku zahájeniu komunikácie, na ktorú odpovedá, potvrdzovacou metódou ACK a hlásením 200 OK, čo znamená, že žiadosť o telefónny hovor bola úspešne akceptovaná. Kvalita zvuku počas hovoru bola celkom dobrá, pokiaľ bol hovor uskutočnený blízko USRP. Vzdialovaním mobilnej stanice od USRP zvuk strácal na kvalite.



Obrázok 19: *SMS*

SIP/SDP	854	Request: INVITE sip:2101@127.0.0.1, with session description
SIP	520	Status: 100 Trying
SIP/SDP	942	Request: INVITE sip:IMSI230025900646538@127.0.0.1:5062, with session description
SIP	449	Status: 100 Trying
SIP	449	Status: 100 Trying
SIP	449	Status: 100 Trying
SIP	450	Status: 180 Ringing
SIP	536	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP	450	Status: 180 Ringing
SIP/SDP	857	Status: 200 OK, with session description
SIP	471	Request: ACK sip:IMSI230025900646538@127.0.0.1:5062
SIP/SDP	765	Status: 200 OK, with session description
SIP	404	Request: ACK sip:2101@127.0.0.1

Obrázek 20: *Tel.hovor Wireshark*

Obrázok 21 demonštruje, že sms správa z telefónneho čísla 2102 bola uložená do fronty správ serveru smqueue, kde čaká na doručenie a následovne je úspešne doručená na číslo 2101, čo potvrdzuje hlásenie 200 OK.

SIP	797	Request: MESSAGE sip:smsc@127.0.0.1(RP)
SIP	317	Status: 202 Queued
SIP	830	Request: MESSAGE sip:IMSI230025900646538@127.0.0.1:5062(RP)
SIP	549	Status: 200 OK

Obrázok 21: *SMS Wireshark*

6.2 Realizácia hovoru pripojením na SIP trunk providera

Po spustení Asterisku som si príkazom *sip show peers* overil, či sú všetky potrebné náležitosti v poriadku pripojené k serveru. Následujúci výpis potvrdzuje, že softphone 2222 bol úspešne pripojený a taktiež komunikácia medzi ústredňami cez SIP trunk prebieha v poriadku cez port 5060.

```
Besip*CLI> sip show peers
Name/username    Host            Dyn Forcerport ACL Port Status
2222/2222        158.196.195.187 D          N       5061 OK (6 ms)
siptr/siptr1     158.196.195.187 D          a       5060 OK (6 ms)
2 sip peers [Monitored: 2 online, 0 offline Unmonitored: 0 online, 0
offline]
```

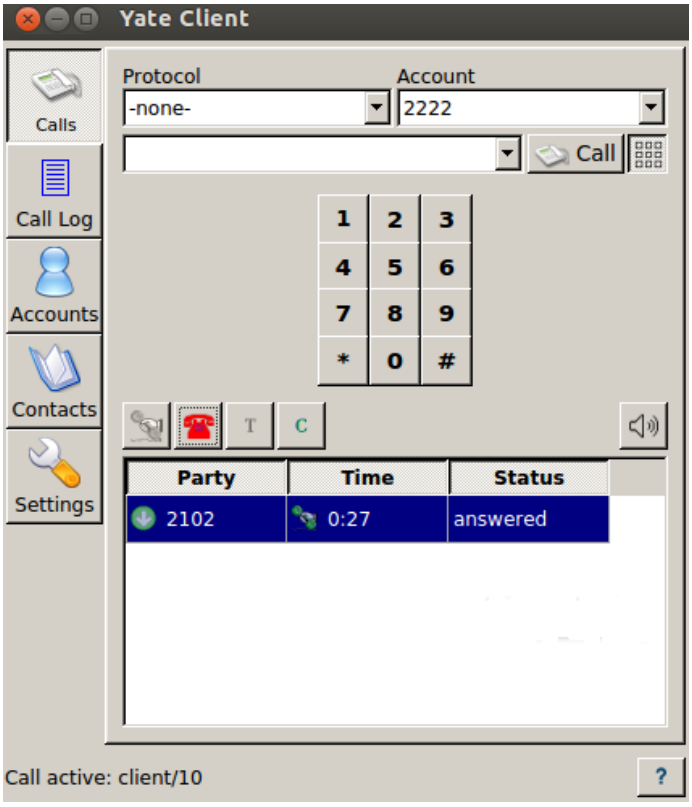
Teraz by už malo byť splnené všetko potrebné pre realizáciu hovoru prostredníctvom SIP trunku. Obrázok 22 poukazuje na to, že po vytočení čísla 2102 na softphone Yate Client, bola poslaná INVITE žiadosť o prijatie hovoru na mobilný telefón s číslom 2102, ktorý po sérii informačných správ

Trying a Ringing, úspešne prijíma hovor potvrdzovacou metódou ACK a na server posiela správu 200 OK o úspešnom ukončení žiadosti.

158.196.115.251	158.196.244.158	SIP/SDP	746	Request: INVITE sip:2102@158.196.244.158, with session description
158.196.244.158	158.196.115.251	SIP	502	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP/SDP	1005	Request: INVITE sip:IMSI230025900646538@127.0.0.1:5062, with session description
127.0.0.1	127.0.0.1	SIP	436	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP	436	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP	436	Status: 100 Trying
127.0.0.1	127.0.0.1	SIP	437	Status: 180 Ringing
158.196.244.158	158.196.115.251	SIP	518	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	437	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	437	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	437	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP	437	Status: 180 Ringing
127.0.0.1	127.0.0.1	SIP/SDP	843	Status: 200 OK, with session description
127.0.0.1	127.0.0.1	SIP	458	Request: ACK sip:IMSI230025900646538@127.0.0.1:5062
158.196.244.158	158.196.115.251	SIP/SDP	835	Status: 200 OK, with session description
158.196.115.251	158.196.244.158	SIP	409	Request: ACK sip:2102@158.196.244.158:5060
127.0.0.1	127.0.0.1	SIP	496	Request: BYE sip:2222@127.0.0.1

Obrázok 22: *Tel.hovor Wireshark (SIP trunk)*

Na nasledujúcom obrázku je taktiež možné vidieť, že žiadosť o telefónny hovor z mobilnej stanice s číslom 2102 zo siete OpenBTS, bola úspešne akceptovaná volaným účastníkom s číslom 2222.



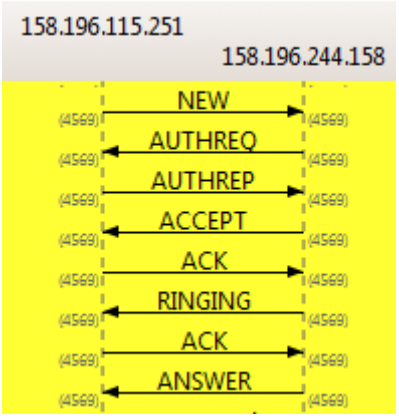
Obrázok 23: *softphone YATE (tel.č 2222)*

6.3 Reliazácia hovoru pripojením na IAX trunk providera

Opäť ako v predchádzajúcom prípade som si príkazom *iax2 show peers* overil, či sa obidve ústredne spojili prostredníctvom IAX trunku a komunikujú cez port 4569, čo potvrdzuje následujúci výpis.

```
Besip*CLI> sip show peers
Name/username Host                 Mask            Port     Status
iaxtr/iaxtr1  158.196.195.187 (D) 255.255.255.255 4569 (T)(E)OK(7ms)
1 iax2 peers [1 online, 0 offline, 0 unmonitored]
```

Na obrázku 23 je možné vidieť, že celý proces začína zriadením hovorového kanálu, kedy volajúce zariadenie z IP 158.196.195.244 pošle správu NEW, v ktorej sú obsiahnuté informácie o adresách. Správou AUTHREQ žiada volaná strana o autorizáciu, na ktorú volajúci odpovedá správou AUTHREP, v ktorej sú obsiahnuté žiadané autorizačné informácie. Server s IP 158.196.244.158 správu spracoval, autorizačné údaje sú správne a informuje volajúceho správou ACCEPT, ktorý spätne informuje o jej prijatí správou ACK. Volaná strana pošle správu RINGING, ktorá signalizuje zvonenie telefónu na opačnej strane. Vo chvíli keď softphone na strane serveru príjme hovor, spustí sa hovorový kanál, čo signalizuje správa ANSWER poslaná volajúcej strane. Na obrázku je taktiež možné vidieť, že komunikácia medzi ústredňami naozaj prebieha prostredníctvom portu 4569.



Obrázok 24: *IAX trunk Wireshark*

Komunikáciu mimo IAX trunku prostredníctvom SIP protokolu zachycuje obrázok 24. Po vytočení telefónneho čísla softphonu 2222 na mobilnom telefóne v sieti OpenBTS, je poslaná správa INVITE, na ktorú softphone po úspešnom prijatí hovoru odpovedá správou ACK. Následne je na číslo 2102 predstavujúce volajúci mobilný telefón poslaná správa BYE, ktorá ho žiada o ukončenie telefónneho hovoru. Telefón s IMSI230025900646538 zasiela správu 200 OK o úspešnom ukončení hovoru.

127.0.0.1	127.0.0.1	SIP/SDP	854	Request: INVITE sip:2222@127.0.0.1 |
127.0.0.1	127.0.0.1	SIP	521	Status: 100 Trying |
158.196.244.158	158.196.115.251	SIP/SDP	1079	Request: INVITE sip:2222@158.196.115.251:53419 |
158.196.115.251	158.196.244.158	SIP	394	Status: 100 Trying |
158.196.115.251	158.196.244.158	SIP	509	Status: 180 Ringing |
127.0.0.1	127.0.0.1	SIP	537	Status: 180 Ringing |
158.196.115.251	158.196.244.158	SIP/SDP	749	Status: 200 OK |
158.196.244.158	158.196.115.251	SIP	466	Request: ACK sip:2222@158.196.115.251:53419 |
127.0.0.1	127.0.0.1	SIP	405	Request: ACK sip:2222@127.0.0.1 |
158.196.244.158	158.196.115.251	SIP	605	Request: OPTIONS sip:2222@158.196.115.251:53419 |
158.196.115.251	158.196.244.158	SIP	388	Status: 100 Trying |
158.196.115.251	158.196.244.158	SIP	456	Status: 200 OK |
158.196.244.158	158.196.115.251	SIP	521	Request: BYE sip:2222@158.196.115.251:53419 |

Obrázok 25 : *Tel.hovor Wireshark (IAX trunk)*

7 Záver

Vo svojej bakalárskej práci som sa venoval OpenBTS a jeho integrácii do telekomunikačnej siete. Najskôr som podrobne popísal všetky potrebné naléžitosti pre realizáciu vlastnej softwarovo riešenej GSM siete, ktorú je možné vytvoriť pomocou hardwaru USRP N210. Funkčnosť systému som demonštroval zachytením komunikácie v softwarom nástroji Wireshark pri telefónnom hovore alebo posielaní sms správ medzi dvomi mobilnými telefónmi v sieti OpenBTS. Nakoľko pre registráciu mobilných účastníkov bolo potrebné manuálne modifikovať sqlite3 databázu Asterisku a konfiguračné súbory Asterisk, tak som pre tento scenár napísal program v jazyku C++, ktorý na základe IMSI čísla mobilných staníc upraví všetky potrebné súbory pre uskutočnenie telefónneho hovoru. Následne som rozšíril sieť OpenBTS o pripojenie na SIP alebo IAX trunk providera, ktorého som vytvoril na serveri Besip, čím som umožnil mobilnému telefónu registrovanému v sieti OpenBTS sa dovolať prostredníctvom internetu prakticky komukoľvek, kto bude zaregistrovaný k serveru. Pre obidva scenáre použitia som komunikáciu zachytil taktiež prostredníctvom aplikácie Wireshark a jednotlivé výsledky zahrnul do svojej bakalárskej práce.

V praktickej realizácii nedochádzalo k veľkým komplikáciam. Nakoľko open-source projekt OpenBTS je stále v štadiu vývoja a mnoho vecí ešte nefunguje tak ako má, tak je nestabilné a sporadicky dochádzalo k výpadkom siete. Ostatné komponenty ako je sipatuhserve alebo smqueue fungovali bezproblémovo . Aj keď prioritou open-source projektu OpenBTS je jeho integrácia do telekomunikačnej siete, videl by som využitelnosť jeho hlavného nástroja USRP N210 aj v iných scenároch ako je realizácia GSM siete. Pre jeho schopnosť zachytávať IMSI čísla by sa napríklad dal využiť ako turniket, kedy by osobe prechádzajúcej turniketom bolo zachytené číslo mobilného telefónu, ktoré by bolo následne porovnané s IMSI databázou, na základe ktorej by bol osobe vstup povolený alebo nepovolený, čím by sa eliminovala potreba vstupeniek. Napríklad taký vstup do veľkých továrni, kde každé ráno sú robotníci povinný ukázať svoje povolenie pre vstup. Opäť by sa to dalo vyriešiť integráciou USRP, ktorý by kontroloval, či je vstupujúci do fabriky registrovaný alebo nie. Podobných scenárov by sa dalo vymyslieť určite mnoho, ale to by už mohlo byť predmetom na vypracovanie pre iné práce.

8 Použitá literatura

[1] ŽALUD, Václav. *Moderní radioelektronika*. 1. vyd. Praha: BEN, 2000, 656 s. ISBN 80-86056-47-3.

[2] RICHTR, Tomáš. *Technologie pro mobilní komunikaci* [online]. 2002 [cit. 2013-04-05]. Dostupný z WWW: <http://tomas.richtr.cz/mobil/bunk-gsm.htm>.

[3] HANUS, Stanislav. *Bezdrátové a mobilní komunikace*. 1. vyd. Brno : VUT v Brně, 2003. 134s. ISBN 80-214-1833-8.

[4] DUDEK, Ondřej. *Struktura GSM*. 2003. 6 s. České vysoké učení technické v Praze, Vedoucí semestrální práce Karel Mikuláštík. Dostupný z WWW:<radio.feld.cvut.cz/personal/mikulak/>.

[5] HEINE, Gunnar: *GSM Networks: Protocols, Terminology and Implementation*. Boston / London: Artech House Publishers, 1999. 416 s. ISBN: 0-89006-471-7

[6] HANUS, Stanislav, FENCL, Josef, ŠTENCEL, Vít. *Bezdrátové a mobilní komunikace II.* 1. vyd. Brno : Vysoké učení technické v Brně, 2005. 171 s. ISBN 80-214-2817-1

[7] BURGESS, D. A, SAMRA H. *The Open BTS Project – an opensource GSM base station.* September 2008. [Online]. Dostupné z: http://www.ahzf.de/itstuff/papers/ OpenBTSProject.pdf

[8] COOPER, Thomas. *Integration of Open-Source GSM Networks*. [Online]. Dostupné z: http://scholar.lib.vt.edu/theses/available/etd-05082012-141540/unrestricted/ Cooper_TA_T_ 2012.pdf

[9] Project Osmocom. [Online]. Dostupné z http://bb.osmocom.org/trac/wiki/ProjectRationale

[10] Welcome to Osmocom OpenBSC. [Online]. Dostupné z: http://openbsc.osmocom.org

[11] OsmocomBB. [Online]. Dostupné z: http://bb.osmocom.org

[12] Libosmocore. [Online]. Dostupné z: http://bb.osmocom.org/trac/wiki/libosmocore

[13] Libosmo-Abis. [Online]. Dostupné z: http://openbsc.osmocom.org/trac/wiki/libosmo-abis 48

[14] OsmoSGSN. [Online]. Dostupné z: http://openbsc.osmocom.org/trac/wiki/osmo-sgsn

[15] OpenGGSN. [Online]. Dostupné z: http://openbsc.osmocom.org/trac/wiki/OpenGGSN

[16] Range Networks. [Online]. Dostupné z: http://www.rangenetworks.com

[17] OpenBTS. [Online]. Dostupné z: http://gnuradio.org/redmine/projects/gnuradio/wiki/ OpenBTSBackground

[18] LEIF MADSEN, Jim Van Meggelen, Leif MADSEN a Jared SMITH. *Asterisk: the definitive guide*. 3rd ed. Sebastopol, CA: O'Reilly Media, Inc, 2007, 574 s. ISBN 978-059-6517-342.

[19] KEMETMULLER Ch., SEEGER M., BAIER H., BUSCH Ch. *Manipulating Mobile Devices with a private GSM Base Station - a Case Study*. [online]. Dostupné z: https://www.fbi.h-da.de/fileadmin/gruppen/FG-IT-Sicherheit/Publikationen/2010/Kemetmueller_Seeger_INC2010.pdf

[20] NOVOTNÝ, V. *Komunikační prostředky mobilních sítí* Brno: Vysoké učení technické v Brně, Fakulta elektroniky a komunikačních technologii, 2006 - 92s.

[21] VAN MEGGELEN, J., MADSEN L., SMITH J. *Asterisk: The Future of Telephony*. O'Reilly, ISBN 978-0-596-51048-0, Sebastopol, USA, 2007.

[22] Ettus Research: A national instruments company. ETTUS RESEARCH. *http://www.ettus.com* [online]. 2014 [cit. 2013-05-03]. Dostupné z: http://www.ettus.com

[23] Úvod do VoIP. MB DATA [online]. [cit. 2014-04-15]. Dostupné z: http://www.mbdata.cz/uvoddovoip.htm#_Využití_IP_telefonie

Zoznam príloh

Príloha A: sip.conf na strane klienta XLVII
Príloha B: extensions.conf na strane klienta 48
Príloha C: iax.conf na strane klienta L
Príloha D: sip.conf na strane serveru LI
Príloha E: extensions.conf na strane serveru LIII
Príloha F: iax.conf na strane serveru LIV
Príloha G: Registrácia astprg.cpp LV
Príloha H: makefile pre kompiláciu astprg.cpp LXIII

Súčasťou BP je CD.

Adresárová štruktúra priloženého CD:

- Registrácia/astprg.cpp
- Registrácia/sqlite3.h
- Registrácia/sqlite3.c
- Registrácia/Makefile
- Registrácia/sqlite3.db
- Konfigurácia/sip.conf
- Konfigurácia/extensions.conf
- Konfigurácia/iax.conf
- Konfigurácia/sip-server.conf
- Konfigurácia/extensions-server.conf
- Konfigurácia/iax-server.conf
- BP/BEH032.pdf

Príloha A: sip.conf na strane klienta

```
[general]
port=5060
bandwidth=low
jitterbuffer=yes

tlsenable=yes              ;povolenie TLS
tlsbindaddr=0.0.0.0        ;pripojenie TLS k IP
tlscertfile=/etc/asterisk/keys/siptr1.pem  ;cesta k certifikátu
tlsdontverifyserver=yes    ;overenie či sa Asterisk správa jako
klient
tlscipher=DES-CBC3-SHA     ;šifrovací algoritmus
tlsclientmethod=tlsv1      ; verzia TLS/SSL

register=>tls://siptr1:trunk@158.196.244.158:5061/siptr

tos_sip=cs3
tos_audio=ef
tos_video=af41
tos_text=af41
cos_sip=3
cos_audio=5
cos_video=4
cos_text=3
maxexpiry=3600
minexpiry=60
defaultexpiry=3600
dynamic_exclude_static=yes
use_q850_reason=yes
rtptimeout=60
rtpholdtimeout=300
allowoverlap=no
```

```
disallow=all
allow=gsm
allow=alaw
allow=g711
relaxdtmf=yes
dtmfmode=auto

[siptr1]
type=friend
host=dynamic
trunk=yes
username=siptr
secret=trunk
qualify=yes
context=internal
dtmfmode=rfc2833

[IMSI230025900646538]
callerid=2102
type=friend
qualify=yes
host=dynamic
context=remote
username=IMSI230025900646538

[IMSI230026700507478]
callerid=2101
type=friend
qualify=yes
host=dynamic
context=remote
username= IMSI230026700507478
```

Príloha B: extensions.conf na strane klienta

```
[globals]

[general]

autofallthrough=yes

[default]

[sip]

include => internal

include => remote

[outbound-trunk]
exten => _N.,1,Answer()

[internal]

exten => _N.,1,Set(Name=${ODBC_SQL(select dial from dialdata_table
where exten = \"${EXTEN}\")})

exten => _N.,n,GotoIf($["${Name}" = ""] ?outbound-trunk,${EXTEN},1)

exten => _N.,n,Set(IPAddr=${ODBC_SQL(select ipaddr from sip_buddies
where name = \"${Name}\")})

exten => _N.,n,GotoIf($["${IPAddr}" = ""] ?outbound-runk,${EXTEN},1)

exten => _N.,n,Dial(SIP/${Name}@${IPAddr}:5062)


[remote]

;(Pravidlá pre spojenie mobilných telefónov v sieti OpenBTS)

exten => 2101,1,Dial(SIP/IMSI230026700507478@127.0.0.1:5062)

exten => 2102,1,Dial(SIP/IMSI230025900646538@127.0.0.1:5062)

;(Pravidlá pre spojenie prostredníctvom IAX trunku)

exten => 2222,1,Dial(SIP/iaxtr1/${EXTEN})

exten => 2222,n,Congestion()

exten => 2222,n,Hangup()

;(Pravidlá pre spojenie prostredníctvom SIP trunku)

exten => 2222,1,Dial(SIP/siptr1/${EXTEN})

exten => 2222,n,Congestion()

exten => 2222,n,Hangup()

[incoming]

include => internal
```

Príloha C: iax.conf na strane klienta

```
[general]
autokill=yes
port=4569
bandwidth=low
disallow=all
jitterbuffer=yes
calltokenoptional=0.0.0.0/0.0.0.0
maxregexpire=130
encryption=yes
disallow=all
allow=alaw
allow=gsm
allow=g711

register=>iaxtr:trunk@158.196.244.158:4569

[iaxtr1]
type=friend
host=dynamic
trunk=yes
secret=trunk
username=iaxtr
qualify=yes
context=internal
deny=0.0.0.0/0.0.0.0
permit=158.196.244.158/255.255.255.255
```

Príloha D: sip.conf na strane serveru

```
[general]
port=5060
bandwidth=low
jitterbuffer=yes
tlsenable=yes
tlsbindaddr=0.0.0.0
tlscertfile=/etc/asterisk/keys/siptr.pem
tlsdontverifyserver=yes
tlscipher=DES-CBC3-SHA
tlsclientmethod=tlsv1

register=>tls://siptr:trunk@158.196.115.251:5061/siptr1

tos_sip=cs3
tos_audio=ef
tos_video=af41
tos_text=af41
cos_sip=3
cos_audio=5
cos_video=4
cos_text=3

maxexpiry=3600
minexpiry=60
defaultexpiry=3600
dynamic_exclude_static=yes
use_q850_reason=yes
rtptimeout=60
use_q850_reason=yes
rtptimeout=60
rtpholdtimeout=300
```

```
autocreatepeer=yes
allowoverlap=no
disallow=all
allow=gsm
allow=alaw
allow=g711
relaxdtmf=yes
dtmfmode=auto

[siptr]
type=friend
host=dynamic
trunk=yes
secret=trunk
username=siptr1
context=internal
dtmfmode=rfc2833

[2222]
callerid=2222
qualify=yes
type=friend
host=dynamic
context=remote
username=2222
```

Príloha E: extensions.conf na strane serveru

```
[sip]
include => internal
include => remote

[internal]
exten => 2222,1,NoOp()
exten => 2222,n,Dial(SIP/${EXTEN},30)
exten => 2222,n,Hangup()

[remote]
(Pravidlá pre spojenie prostredníctvom IAX trunku)
exten => 2102,1,NoOp()
exten => 2102,n,Dial(IAX2/iaxtr/${EXTEN})
exten => 2102,n,Hangup()

(Pravidlá pre spojenie prostredníctvom SIP trunku)
;exten => 2102,1,NoOp()
;exten => 2102,n,Dial(SIP/siptr/${EXTEN})
;exten => 2102,n,Hangup()

[incoming]
include => remote
```

Príloha F: iax.conf na strane serveru

```
[general]
autokill=yes
bindport=4569
disallow=all
1jitterbuffer=yes
calltokenoptional=0.0.0.0/0.0.0.0
requierecalltoken=no
maxcallnumbers=16381
maxregexpire=130
encryption=yes

register=>iaxtr1:trunk@158.196.115.251
disallow=all
allow=gsm
allow=alaw
allow=g711

[iaxtr]
type=friend
host=dynamic
trunk=yes
secret=trunk
username=iaxtr1
context=internal
deny=0.0.0.0/0.0.0.0
permit=158.196.115.251/255.255.255.255
```

Príloha G: Registrácia astprg.cpp

```
//knižnice potrebné pre správnu funkčnosť programu
#include <cstdlib>
#include <cstring>
#include <cstdio>
#include <string>
#include <iostream>
#include <sstream>
#include <fstream>

//program potrebuje pre svoju funkčnosť hlavičkový súbor sqlite3.h
extern "C"
{
     #include "sqlite3.h"
}

// konštanta dbPath deklaruje cestu k súboru sqlite3.db
const std::string dbPath =/var/lib/asterisk/sqlite3dir/sqlite3.db";

/*konštanta asteriskPath definuje cestu do hlavného adresára
Asterisku */
const std::string asteriskPath = "/etc/asterisk";

//cesta k súboru sip.conf
const std::string sipConfPath = asteriskPath + "/sip.conf";

//cesta k súboru extensions.conf
const std::string extensionsConfPath=asteriskPath+/extensions.conf";

/* context [remote] v súbore extensions.conf, do ktorého bude
vložené volacie pravidlo pre vložené IMSI číslo */
const std::string extensionsConfSection = "[remote]";

//deklarácie funkcii
bool searchForImsi(sqlite3 *db, std::string imsi);
int generateId(sqlite3 *db);
std::string generatePhoneNumber(sqlite3 *db);
void insertRecordToDB(sqlite3 *db, int id, std::string number,
std::string imsi);
void insertRecordToDialdata(sqlite3 *db, int id, std::string number,
std::string imsi);
void insertRecordToSipBuddies(sqlite3 *db, int id, std::string
number, std::string imsi);
void createRecordInSipConf(std::string number, std::string imsi);
void createRecordInExtensionsConf(std::string number, std::string
imsi);
```

```
//hlavný program
int main(int argc, char *argv[])
{
      //program vyžaduje ako druhý argument IMSI číslo
      if (argc < 2 || argc > 2)
      {
            //argument nebol zadaný
            std::cout << "Invalid number of arguments: must be 2" <<
            std::endl;
            return 1;
      }
      //IMSI číslo bolo nájdené
      std::string imsi(argv[1]);

      //otvorenie databáze
      std::cout << "Opening database...";
      sqlite3 *db;
      int result = sqlite3_open(dbPath.c_str(), &db);

      //nepodarilo sa otvoriť databázu
      if (result != SQLITE_OK)
      {
            return 1;
      }

      //databáza bol úspešne otvorená
      std::cout << "OK" << std::endl;

      //funkcia pre zistenie, či už sa dané IMSI v DB nenachádza
      bool imsiFound = searchForImsi(db, imsi);

      //IMSI číslo sa v DB nenašlo
      if (!imsiFound)
      {
            /* IMSI sa vygeneruje ID číslo a telefónne číslo,
            následne sú údaje vložené do sqlite3.db, sip.conf a
            extensions.conf */
            std::cout << "IMSI not used, creating new record." <<
            std::endl;
            int id = generateId(db);
            std::string number = generatePhoneNumber(db);
            insertRecordToDB(db, id, number, imsi);
            createRecordInSipConf(number, imsi);
            createRecordInExtensionsConf(number, imsi);
      }
```

```
	/* ak sa IMSI našlo v DB je ignorované, súbory sa nebudú
	aktualizovať */
	else
	{
		std::cout << "IMSI is used, ignoring." << std::endl;
	}
	//zatvorenie databáze
	std::cout << "Closing DB...";
	sqlite3_close(db);
	std::cout << "OK" << std::endl;
	return EXIT_SUCCESS;
}

/* bool searchForImsi je funkcia, ktorá slúži pre nájdenie IMSI
čísla v databáze Asterisku sqlite3.db */
bool searchForImsi(sqlite3 *db, std::string imsi)
{
	sqlite3_stmt *statement;
	const char *query = "SELECT COUNT(*) FROM dialdata_table WHERE
	dial = ?;";
	sqlite3_prepare_v2(db, query, strlen(query), &statement, NULL);
	sqlite3_bind_text(statement, 1, imsi.c_str(),-1,SQLITE_STATIC);
	sqlite3_step(statement);
	int founded = sqlite3_column_int(statement, 0);
	sqlite3_finalize(statement);
	return founded > 0;
}

/* funkcia pre vygenerovanie ID čísla pre vkládané IMSI, ktorá nájde
v databáze najväčšie existujúce ID číslo a pripočíta k nemu 1 */
int generateId(sqlite3 *db)
{
	sqlite3_stmt *statement;
	const char *query = "SELECT MAX(id) FROM dialdata_table;";
	sqlite3_prepare_v2(db, query, strlen(query), &statement, NULL);
	sqlite3_step(statement);
	int highestId = sqlite3_column_int(statement, 0);
	int newId = highestId + 1;
	sqlite3_finalize(statement);
	return newId;
}
```

```
/*funkcia pre vygenerovanie telefónneho čísla, ktorá podobne jako
predchádzajúca funkcia, nájde v databáze najväčšie telefónne číslo a
pripočíta k nemu 1 */
std::string generatePhoneNumber(sqlite3 *db)
{
      sqlite3_stmt *statement;
      const char *query = "SELECT MAX(exten) FROM dialdata_table;";
      sqlite3_prepare_v2(db, query, strlen(query), &statement, NULL);
      sqlite3_step(statement);
      int highestPhoneNumber = sqlite3_column_int(statement, 0);
      std::stringstream ss;
      ss << (highestPhoneNumber + 1);
      std::string newPhoneNumber(ss.str());
      sqlite3_finalize(statement);
      return newPhoneNumber;
}

/* vloženie údajov do databáze, funkcia má štyri parametre
1.sqlite3 *db   - konštanta definujúca cestu k súboru sqlite3.db
2.int id        - vygenerované ID číslo
3.string number - vygenerované telefónne číslo
4.string imsi   - IMSI číslo vložené při spúšťaní programu */

void insertRecordToDB(sqlite3 *db, int id, std::string number,
std::string imsi)
{
      std::cout << "Inserting to db: id(" << id << "), number(" <<
      number << "), imsi(" << imsi << ")" << std::endl;

      //funkcia pre vloženie parametrov do dial_data tabuľky
      insertRecordToDialdata(db, id, number, imsi);

      //funkcia pre vloženie parametrov do sip_buddies tabuľky
      insertRecordToSipBuddies(db, id, number, imsi);
}

/* funkcia pre vloženie ID čísla, telefónneho čísla a IMSI čísla do
dial_data tabuľky */
void insertRecordToDialdata(sqlite3 *db, int id, std::string number,
std::string imsi)
{
      std::cout << "Inserting to dialdata_table...";
      sqlite3_stmt *statement;
      const char *query = "INSERT INTO dialdata_table (id, exten,
      dial) VALUES (?, ?, ?);";
      sqlite3_prepare_v2(db, query, strlen(query), &statement, NULL);
      sqlite3_bind_int(statement, 1, id);
```

```
	sqlite3_bind_text(statement, 2, number.c_str(), -1,
	SQLITE_STATIC);
	sqlite3_bind_text(statement, 3, imsi.c_str(), -1,
	SQLITE_STATIC);

	int result = sqlite3_step(statement);
	if (result != SQLITE_DONE)
	{
		std::cout << "Error!" << std::endl;
	}
	else
	{
		std::cout << "OK" << std::endl;
	}
	sqlite3_finalize(statement);
}


/*funkcia pre vloženie parametrov do sip_buddies tabuľky, v
konštante query sú definované data, ktoré majú byť spolu s IMSI
číslo vložené do databáze,kde je nutné ho zaregistrovať ab IMSI
číslo mohlo realizovať telefónne hovory. Otázniky v konštante query
sú potom pomocou funkcie sqlite_bind nahradené postupne ID číslom,
IMSI číslom, telefónnym číslom a nakoniec opäť IMSI číslom */
void insertRecordToSipBuddies(sqlite3 *db, int id, std::string
number, std::string imsi)
{
	std::cout << "Inserting to SIP_BUDDIES...";
	sqlite3_stmt *statement;
	const char *query = "INSERT INTO \"SIP_BUDDIES\" VALUES \
	(?, ?,
	internal','allowed_not_screened',NULL,NULL,NULL,NULL,NULL,
	\NULL,'dynamic','no','friend',NULL,NULL,NULL,?,'0.0.0.0','info'
	,\NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,NULL,'all',
	'gsm',\NULL,'127.0.0.1',5062,?,NULL,NULL,NULL,NULL,NULL,NULL,NU
	LL,NULL,\NULL,NULL,'yes','yes','yes','no',NULL,'no',NULL,'yes',
	'accept',1800,\90,'uas',NULL,NULL,NULL,'yes',500,NULL,120,NULL,
	NULL,0,NULL,0,NULL,\'yes','no',NULL,NULL,NULL,NULL,0,0,NULL,NUL
	L,NULL,NULL,1,NULL,0,NULL);";

	sqlite3_prepare_v2(db, query, strlen(query), &statement, NULL);
	sqlite3_bind_int(statement, 1, id);
	sqlite3_bind_text(statement, 2, imsi.c_str(), -1,
	SQLITE_STATIC);
	sqlite3_bind_text(statement, 3, number.c_str(), -1,
	SQLITE_STATIC);
```

```
	sqlite3_bind_text(statement, 4, imsi.c_str(), -1,
	SQLITE_STATIC);
	//kontrola či operácia prebehla v poriadku
	int result = sqlite3_step(statement);
	if (result != SQLITE_DONE)
	{
		//operácia neprebehla v poriadku
		std::cout << "Error!" << std::endl;
	}
	else
	{
		//vloženie údajov do databáze prebehlo v poriadku
		std::cout << "OK" << std::endl;
	}

	sqlite3_finalize(statement);
}

/* vytvorenie konfigurácie pre IMSI číslo a jej následné vloženie do
konfiguračného súboru Asterisku sip.conf */
void createRecordInSipConf(std::string number, std::string imsi)
{
	// vzor
	/*
		[IMSI230025900646538]  ;Moj telefon
		callerid=2102          ;programom náhodne pridelené číslo
		canreinvite=no
		type=friend
		context=remote
		allow=gsm
		host=dynamic
		dtmfmode=RFC2833
	*/

	std::ofstream outfile;
	outfile.open(sipConfPath.c_str(), std::ios_base::app);

	//vytvorenie kontextu pre IMSI v sip.conf
	outfile
	<< "[" << imsi << "]" << std::endl
	<< "callerid=" << number << std::endl
	<< "canreinvite=no" << std::endl
	<< "type=friend" << std::endl
	<< "context=remote" << std::endl
	<< "allow=gsm" << std::endl
	<< "host=dynamic" << std::endl
	<< "dtmfmode=RFC2833" << std::endl
```

```
	<< std::endl;

	// zatvorenie súboru
	outfile.close();

	// kontrola či operácia prebehla úspešne
	if (outfile.fail())
	{
		// operácia neprebehla v poriadku
		std::cout << "Problem, can't write to file " <<
		sipConfPath << std::endl;
	}
	else
	{
		/* operácia prebehla v poriadku, súbor sip.conf bol
		aktualizovaný*/
		std::cout << sipConfPath << " updated." << std::endl;
	}
}

//funkcia pre pridanie záznamu do dialplánu extensions.conf
void createRecordInExtensionsConf(std::string number, std::string
imsi)
{
	// otvorenie pomocného súboru
	std::ofstream outfile;

	//cesta k súboru
	std::string tempPath = extensionsConfPath + "~";

	outfile.open(tempPath.c_str(), std::ios_base::ate |
	std::ios_base::trunc);

	// prečítanie údajov z originálneho súboru
	std::ifstream infile;
	infile.open(extensionsConfPath.c_str());

	//načítanie súboru
	while (true)
	{
		std::string line;
		getline(infile, line);
		if (infile.eof())
		{
			break;
		}
		outfile << line << std::endl;
```

```
            std::cout << "---" << line << std::endl;

            // nájdenie kontextu remote v extensions.conf
            if (line.compare(extensionsConfSection) == 0)
            {

            /* pokiaľ bol context nájdený, zapíše sa do neho
            nasledujúca konfigurácia */
            outfile<< "exten => " << number << ",1,Dial(SIP/" << imsi
            << "@127.0.0.1:5062)" << std::endl;
            }
      }

      //kontrola či operácia prebehla úspešne
      if (outfile.fail())
      {
            //pokiaľ nie, vypíš chybovú hlášku
            std::cout << "Problem, can't write to file " <<
            extensionsConfPath << std::endl;
      }
      else
      {

            /* vloženie do súboru extensions.conf bolo úspešné, súbor
            bol aktualizovaný */
            std::cout << extensionsConfPath << " updated." <<
            std::endl;
      }
      /* zatvorenie súboru extensions.conf a pomocného súboru
      extensions.conf~ */
      infile.close();
      outfile.close();

      // premenovanie pomocného súboru na originál
      int result = std::rename(tempPath.c_str(),
      extensionsConfPath.c_str());

      if (result != 0)
      {
            /* nie je možné premenovať súbor na originál
            extensions.conf */
            std::cout << "Can't rename extensions.conf temporal file
            to original." << std::endl;
      }
}
```

Príloha H: makefile pre kompiláciu astprg.cpp

```
all : astprg
astprg : astprg.cpp sqlite3.o
     g++ -o astprg astprg.cpp sqlite3.o -lpthread -ldl
sqlite3.o : sqlite3.c
     gcc -o sqlite3.o -c sqlite3.c

clean :
     rm -f astprg

.PHONY: all, clean
```