VŠB TECHNICKÁ UNIVERZITA OSTRAVA | FAKULTA ELEKTROTECHNIKY A INFORMATIKY

Nástroje pro simulaci technologie LTE

Tools for LTE technology simulation

Bc. Dominik Holaň

Diplomová práce

Vedoucí práce: Ing. Libor Michalek, Ph.D.

Ostrava, 2022

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

Zadání diplomové práce

Student: **Bc. Dominik Holaň**

Studijní program: N2647 Informační a komunikační technologie

Studijní obor: 2612T059 Mobilní technologie

Téma: Nástroje pro simulaci technologie LTE
Tools for LTE Technology Simulation

Jazyk vypracování: čeština

Zásady pro vypracování:

Cílem práce je popsat vlastnosti SimuLTE Framework pro simulační nástroj OMNet++, popsat vlastnosti modulu LTE pro simulační nástroj NS-3, srovnat je a navrhnout komplexní úlohy pro odborný předmět.
1. Popište možnosti SimuLTE Framework pro simulační nástroj OMNet++ a modulu LTE pro simulační nástroj NS-3.
2. Navrhněte a realizujte komplexní simulační příklady využívající oba moduly. Oba příklady se budou zabývat podobnou problematikou s cílem srovnat oba nástroje z pohledu možností, výkonnosti a efektivity řešení.
3. Otestujte tato řešení, celý postup zdokumentujte formou zadání laboratorních cvičení do odborného předmětu.
4. Dosažené výsledky vyhodnoťte s ohledem na efektivitu práce s nástroji, náročnosti a rozsahu funkcí.

Seznam doporučené odborné literatury:

[1] VIRDIS, Antonio; STEA, Giovanni; NARDINI, Giovanni. SimuLTE-A modular system-level simulator for LTE/LTE-A networks based on OMNeT++. In: *2014 International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH)*. IEEE, 2014. p. 59-70.

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

Vedoucí diplomové práce: **Ing. Libor Michalek, Ph.D.**

Datum zadání: 01.09.2021

Datum odevzdání: 30.04.2022

prof. Ing. Miroslav Vozňák, Ph.D.
vedoucí katedry

prof. Ing. Jan Platoš, Ph.D.
děkan fakulty

Abstrakt

Cílem této diplomové práce je popsat vlastnosti frameworku SimuLTE pro simulační nástroj OMNeT++, popsat vlastnosti modulu LTE pro simulační nástroj ns-3, srovnat je a navrhnout laboratorní úlohy pro odborný předmět. První část diplomové práce se zabývá popisem vlastností SimuLTE frameworku. Druhá část práce se zabývá popisem vlastností modulu LTE pro simulační nástroj ns-3. V praktické části následuje návrh komplexních simulačních příkladů, které využívají oba moduly. Simulační příklady se zabývají podobnou problematikou, s cílem porovnat jednotlivé nástroje z pohledu různých možností v dané problematice. Součástí příkladů je vyhodnocení dosažených výsledků. Poslední část diplomové práce se zabývá srovnáním jednotlivých nástrojů z hlediska efektivity, výkonnosti a rozsahu možností.

Klíčová slova

framework, SimuLTE, OMNeT++, ns-3, LTE

Abstract

The aim of this diploma thesis is to describe the properties of the SimuLTE framework for the simulation tool OMNeT++, to describe the properties of the LTE module for the simulation tool ns-3, compare them and design laboratory tasks for a technical subject. The first part of the diploma thesis deals with the description of the properties of the SimuLTE framework. The second part deals with the description of the properties of the LTE module for the simulation tool ns-3. In practical part is the design of complex simulation examples that use both modules. Simulation examples deal with similar issues, in order to compare the various tools in terms of different options. Part of the examples is the evaluation of the achieved results. The last part of the diploma thesis deals with the comparison in terms of efficiency, performance and the range of possibilities of individual tool.

Keywords

framework, SimuLTE, OMNeT++, ns-3, LTE

Poděkování

Velice rád bych poděkoval Ing. Liboru Michalkovi, Ph.D. za odbornou pomoc, rady a konzultace při vytváření této diplomové práce. Rád bych také poděkoval své rodině za nesmírnou podporu v průběhu vysokoškolského studia.

Obsah

Seznam použitých zkratek

Zkratka	Anglický význam	Český význam
3GPP	3rd Generation Partnership Project	Partnerský projekt třetí generace
ACK	Acknowledgement	Potvrzení
AM	Acknowledgment mode	Potvrzovaný režim
AMC	Adaptive modulation coding	Adaptivní modulační kódování
API	Application interface	Aplikační rozhraní
BAT	Blind average throughput [bit/s]	Slepá průměrná propustnost [bit/s]
BGP	Border gateway protocol	Hraniční protocol
BLER	Block error rate [-]	Chybovost bloků [-]
CBR	Constant bit rate [bit/s]	Konstantní bitová rychlost [bit/s]
CID	Cell ID	ID buňky
CMDENV	Comand line environment	Prostředí příkazového řádku
CPU	Control processing unit	Procesor
CQA	Channel and QoS average	Průměr kanálu a QoS
CQI	Channel quality identifier [-]	Identifikátor kvality kanálu [-]
CSV	Comma separated value	Hodnoty oddělené čárkami
CoMP	Coordinate MultiPoint	Vícebodová koordinace
DL	Downlink	Downlink
DRR	Deficit round robin	Deficitní round robin
D2D	Device to device	Zařízení k zařízení
DCI	Data control indication	Indikace kontroly dat
EPC	Evolved packet core	Jádro sítě LTE
E-UTRA	Evolved universal terrestrial radio access	Evoluční univerzální rádiová přístupová síť
ELENA	Extension for ns-3 LTE module	Rozšíření pro ns-3 LTE modul
ETSI TS	European Telecommunications Standards Institute Technical Specification	Technická specifika Evropského ústavu pro telekomunikační normy
EPS	Evolved packet system	Systém sítě LTE
eNB	Evolved node B	Základnová stanice v LTE
eNodeB	Evolved node B	Základnová stanice v LTE
FDD	Frequency division duplex	Duplex frekvenčního dělení
GUI	Graphical user interface	Grafické uživatelské rozhraní
H-ARQ	Hybrid automatic repeat request	Hybridní automatický opakovaný požadavek
HARQ	Hybrid automatic repeat request	Hybridní automatický opakovaný požadavek
ID	Identifier	Identifikátor
IDE	Integrated development environment	Integrované vývojové prostředí
INET	Internet networking	Internetové síťování
INI	Configuration file	Konfigurační soubor

IP	Internet protocol	Internetový protokol
IPv4	Internet protocol version 4	Internetový protokol verze 4
IPv6	Internet protocol version 6	Internetový protokol verze 6
ISO/OSI	International Standards Organization Open Systems Interconnection	Mezinárodní organizace pro normalizaci Propojení otevřených systémů
LCID	Logical cell id	Logický identifikátor buňky
LEGO	Leg godt (play well)	Leg godt (hraj dobře)
LENA	Ns-3 LTE module	Ns-3 LTE modul
LTE	Long term evolution	Dlouhodobý vývoj
LTE NIC	Long term evolution network interface card	Karta síťového rozhraní dlouhodobého vývoje
MAC	Media access control	Řízení přístupu k médiím
MAX C/I	Maximum carrier over interference [dB]	Maximální přenos k rušení [dB]
MCS	Modulation coding scheme	Modulační kódové schéma
MIMO	Multiple input multiple output	Vícenásobný vstup vícenásobný výstup
MME	Mobility management entity	Entita správy mobility
MT	Maximum throughput [bit/s]	Maximální propustnost [bit/s]
NED	Network description	Popis sítě
NS-3	Network simulator 3	Síťový simulátor 3
NetAnim	Network animator	Animátor sítě
OCS	Online charging system	Online účtovací systém
OFDM	Orthogonal frequency division multiplexing	Ortogonální multiplex s frekvenčním dělením
OMNeT++	Operation and Maintenance New Equipment Training	Provoz a údržba nového trénovacího vybavení
OSPF	Open shortest path first	Na základě nejkratší cesty
PCRF	Policy and Charging Rules Function	Funkce politiky a účtovacích pravidel
PDCP	Packet Data Convergence Protocol	Paketový datový konvergenční protokol
PDN	Public data network	Veřejná datová síť
PDU	Packet datagram unit	Paketová datagramová jednotka
PF	Proportional Fair	Proporčně spravedlivý
PGW	Packet Data Network Gateway	Paketová datová síťová brána
PHY	Physical layer	Fyzická vrstva
PPP	Point to point protocol	Protokol bod-bod
PS	Priority set	Prioritní množina
PSS	Primary synchronization signal	Primární synchronizační signál
PyViz	Python visualizer	Python vizualizér
QAM	Quadrature amplitude modulation	Kvadraturní amplitudová modulace
QTENV	OMNeT++ GUI	Grafické uživatelské rozhraní pro OMNeT++

QoS	Quality of Service	Kvalita služby
RAC	Random access control	Procedura náhodného přístupu
RB	Resource block	Přenosová jednotka LTE
RIP	Routing information protocol	Směrovací informační protokol
RLC	Radio link control	Ovládání rádiového spojení
ROHC	Robust header compression	Komprese robustní hlavičky
RR	Round robin	Každý s každým
RRC	Radio resource control	Správa rádiových prostředků
RSRP	Reference signal received power [dBm]	Přijatý výkon referenčního signálu [dBm]
RSRQ	Reference signal received quality [dB]	Kvalita příjmu referenčního signálu [dB]
RSSI	Received Signal Strength Indicator [dBm]	Indikátor přijaté síly signálu [dBm]
SCTP	Stream control transmission protocol	Přenosový protokol řízení toku
SDU	Service data unit	Servisní datová jednotka
SGW	Serving gateway	Obslužná brána v LTE
SimuLTE	LTE User Plane Simulation Model	Simulační model pro LTE uživatelskou vrstvu
SINR	Signal to interference plus noise ratio [dB]	Poměr signálu k rušení a šumu [dB]
SISO	Single input single output	Jeden vstup, jeden výstup
SM	Saturated mode	Saturační režim
SN	Sequence number	Číslo sekvence
SSS	Secondary synchronization signal	Sekundární synchronizační signál
SU-MIMO	Single user multiple input multiple output	Jeden uživatel, více vstupů více výstupů
TB	Token bank	Tokenová banka
TBFQ	Token bank fair queue	Spravedlivá fronta tokenové banky
TCP	Transmission control protocol	Protokol kontroly přenosu
TDD	Time division duplex	Duplex časového dělení
TM	Transparent mode	Transparentní režim
TTA	Throughput to average [-]	Propustnost k průměru [-]
TTI	Transmission time interval [s]	Přenosový časový interval [s]
UDP	User datagram protocol	Datagramový uživatelský protokol
UE	User equipment	Uživatelské vybavení
UL	Uplink	Uplnik
UM	Unconfirmed mode	Nepotvrzovaný režim
VOIP	Voice over internet protocol	Hlas přes internetový protokol
Wi-fi	Wireless fidelity	Bezdrátová věrnost
XML	Extensible Markup Language	Rozšířitelný značkovací jazyk

Seznam ilustrací a tabulek

Úvod

Simulování sítě je ve výzkumu počítačových sítí technika, při níž počítačový program replikuje chování a architekturu skutečné, reálné sítě. Toho je dosaženo pomocí výpočtu interakcí mezi různými síťovými entitami. Cílem simulátoru je co nejlépe předpovědět chování reálné sítě. Komunikační sítě se staly příliš složitými na to, aby tradiční analytické metody poskytly přesné pochopení chování systému. Z tohoto důvodu se v současné době používají síťové simulátory. Diplomová práce se zabývá dvěma simulátory pro simulaci chování LTE sítí. Konkrétně se jedná o network simulator 3 (ns-3) a jeho modul LTE-EPC Network simulator (LENA), simulátor Objective Modular Network Testbed in C++ (OMNeT++) a jeho framework SimuLTE. Oba frameworky jsou vytvořeny v programovacím jazyce C++, kde v ns-3 simulátoru uživatel vytváří simulace také v jazyce C++ nebo v jazyce Python, v prostředí OMNeT++ je poté tvorba simulačních scénářů založena na vlastním vytvořeném jazyce pro potřeby OMNeT++, který se nazývá NED (Network Description language).

První část diplomové práce se zabývá krátkým popisem vlastností OMNeT++, INET a frameworku SimuLTE. Popis SimuLTE je rozčleněn na popis architektury z hlediska vrstev frameworku, které odpovídají vrstvám LTE protokolového modelu, dále pak popisem v prostředí OMNeT++, vytvářením simulačních scénářů v prostředí SimuLTE, měření statistik, možnosti zajištění náhodných běhů simulací a instalace frameworku.

Druhou částí je popis možností modulu LTE (LENA) pro simulační nástroj ns-3. Hlavní kapitola je rozdělena na stručný popis síťového simulátoru, dále pak popis architektury a popis vrstev modulu LTE odpovídající vrstvám LTE protokolového modelu. Následují možnosti různých vývojových prostředí pro LTE modul, možnosti grafické reprezentace trasovacích zdrojů simulace a ukázka jakým způsobem může uživatel zajistit nahrávání trasovacích zdrojů v simulaci. Poslední části je krátký popis instalace na operačním systému Linux.

Třetí částí diplomové práce je návrh a realizace praktických simulačních příkladů, které využívají oba moduly. Příklady se zabývají podobnou problematikou, s cílem srovnání obou nástrojů z pohledu nabízených možností v dané problematice. Navržené příklady jsou rozděleny na celkem pět podkapitol, které dohromady popisují kompletní postup daného příkladu. Navržené simulační scénáře jsou čtenáři dokumentovány formou zadání laboratorních cvičení do odborného předmětu. Součástí příkladů je i vyhodnocení dosažených výsledků simulací, se zaměřením na specifické vlastnosti a možnosti jednotlivých frameworků.

Poslední část diplomové práce se zabývá srovnáním frameworků z hlediska výkonnosti, efektivity a rozsahu funkcí, které daný framework poskytuje. Pro porovnání frameworků z hlediska výkonnosti byl navržen a vytvořen rozsáhlý, identický simulační scénář s možností nastavení libovolného počtu eNodeB v simulaci. Výkonnost byla sledována z hlediska závislosti výpočetní doby na počtu eNodeB v simulaci, velikosti generovaného provozu v simulaci za sekundu a dále byla ověřena linearita výpočetní doby simulací jednotlivých modulů v závislosti na simulačním čase. Následně byly frameworky srovnány z pohledu efektivity práce a přehledu funkcí, které daný nástroj uživateli poskytuje pro potřeby simulace LTE sítí.

1 SimuLTE framework pro OMNeT++

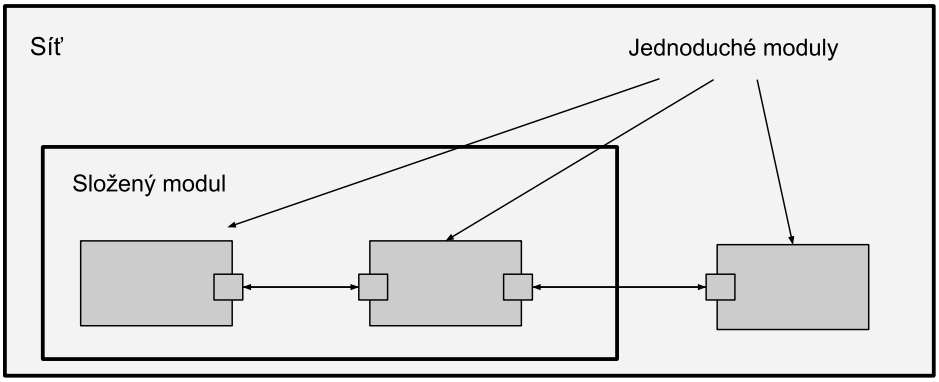
1.1 OMNeT++ Framework

SimuLTE je založený na simulačním nástroji OMNeT++. OMNeT++ může být použit pro modelování více problémů – nejen sítí, jedná se o problémy jako modelování protokolů, multiprocesorů a ostatních distribučních hardwarových systémů, sítí s frontami, drátových, bezdrátových sítí a zjišťování výkonnosti komplexních softwarových systémů. Z tohoto lze vyvodit, že tento simulační nástroj ve své původní podobě nesimuluje nic konkrétního, ale provádí pouze infrastrukturu pro specifické simulační nástroje. Modely jsou sestaveny z opakovaně použitelných komponent nazývaných moduly. Základními stavebními bloky v OMNeT++ jsou tedy moduly, které mohou být jednoduché nebo složené. Dobře napsané moduly jsou opakovatelně použitelné a lze je kombinovat různými způsoby, například jako bloky LEGO. Hloubka vnoření jednotlivých modulů je nekonečná. Moduly mezi sebou komunikují pomocí zpráv, které jsou obvykle zasílány a přijímány pomocí spojení, které spojují jednotlivé brány, takže se ve skutečnosti tváří jako rozhraní. [1][4]

Síť je zvláštní složený modul bez bran do vnějšího okolí, který je nejvýše v hierarchii. Spojení se vyznačují bitovou rychlostí, zpožděním a ztrátovou rychlostí a nemohou obejít hierarchii modulů. Jednoduché moduly implementují chování modelu. Ačkoli každý jednoduchý modul může běžet jako nezávislý, nejběžnějším přístupem je poskytnout jim obslužné rutiny událostí, které jsou volány simulačním jádrem, když moduly obdrží zprávu. Kromě manipulátorů mají jednoduché moduly inicializační funkci a finalizační funkci, která se často používá k zápisu výsledků do souboru na konci simulace. Jádro obsahuje frontu událostí, jejíž události odpovídají zprávám. [1][4]

OMNeT++ umožňuje udržovat samostatnou implementaci, popis a hodnoty parametrů modelu. Implementace (chování) je napsána v C++. Popis (tj. brány, připojení a definice parametrů) je vyjádřen v souborech napsaných v jazyce Network Description (NED). Je zde upřednostňován modulární vývoj umožňující dědičnost popisu modulu i chování. Hodnoty parametrů se zapisují při inicializaci (INI soubory). NED je deklarativní jazyk, který využívá dědičnost, rozhraní a je plně převoditelný (zpět a dále) do XML. NED umožňuje psát parametrické topologie, např. kruh nebo strom proměnlivé velikosti. NED soubory lze editovat textově i graficky (prostřednictvím grafického uživatelského rozhraní, GUI). Soubory INI obsahují hodnoty parametrů, které jsou použity k inicializaci modelu. [1][4]

Pokud jde o automatizaci simulačních pracovních postupů, OMNeT++ odděluje modely (tj. soubory C ++ a NED) od studií. Studie se generují ze souborů INI automatickým vytvářením kartézského součinu všech hodnot parametrů a případně generování replik stejné instance s různými klíči pro generátory náhodných čísel. IDE umožňuje spouštět a spravovat více běhů paralelně a využívat tak více CPU nebo jader, a tím pádem zkrátit dobu provádění simulace. Analýza dat je založena na pravidlech: uživatel musí pouze specifikovat soubory a složky, ze kterých chce extrahovat data, a tzv. recept na filtrování nebo agregaci dat. IDE automaticky aktualizuje a kreslí grafy, při novém spuštění simulace. Výkon OMNeT++ byl srovnáván s jinými simulátory, zejména ns-3. [1][4]



Obrázek 1.1: *Síť, jednoduché a složené moduly [4]*

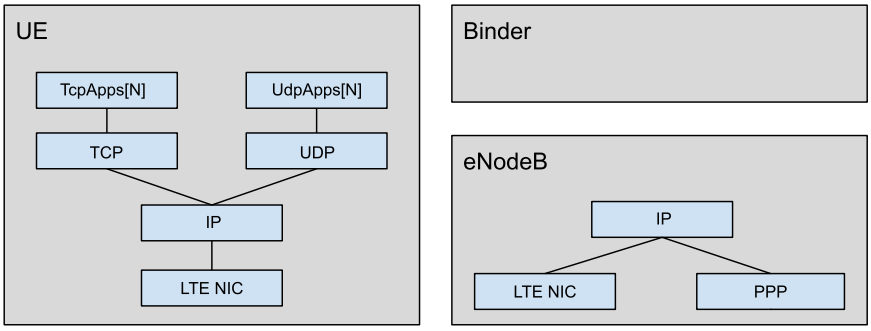
1.2 INET framework

SimuLTE pro svou správnou funkčnost potřebuje INET Framework. INET Framework je open-source modelová knihovna pro OMNeT++. Poskytuje protokoly, agenty a další modely pro výzkumníky a studenty pracující s komunikačními sítěmi. INET je obzvláště užitečný při navrhování a ověřování nových protokolů nebo při zkoumání nových či exotických scénářů. Obsahuje modely pro internetovou vrstvu (TCP, UDP, IPv4, IPv6, OSPF, BGP, RIP), drátové a bezdrátové protokoly linkové vrstvy jako Ethernet, PPP, Wi-fi. Spousty ostatních simulačních frameworků závisí právě na INETu. Jedním z nich je výše zmiňovaný SimuLTE. INET těží z infrastruktury poskytované OMNeT++ také díky využívání služeb poskytovaných simulačním jádrem a knihovnou OMNeT++ (model komponenty, parametrizace, záznam výsledků atd.). To znamená, že modely lze vyvíjet, sestavovat, parametrizovat, spouštět a jejich výsledky vyhodnocovat z OMNeT++ IDE nebo z příkazového řádku. Mezi nejpoužívanější funkce INET frameworku patří: implementace všech vrstev ISO-OSI modelu, IPv4/IPv6, protokoly transportní vrstvy TCP, UDP, SCTP, směrovací protokoly, drátové a bezdrátové rozhraní, široký rozsah aplikačních modelů, emulace sítí. [5]

1.3 SimuLTE framework

Simulační nástroj zvaný SimuLTE umožňuje komplexní výkonové vyhodnocení na úrovni systému sítí LTE a LTE Advanced pro framework OMNeT++. SimuLTE je napsán v C++ a je plně přizpůsobitelný pomocí jednoduchého rozhraní. SimuLTE je open source projekt založený na OMNeT++ a INET frameworku. Nejnovější verze je 1.2.0 a ke své funkci potřebuje OMNeT++ 5.6.2 a INET 4.2.2. Podporuje eNodeB a User Equipment modely, celý protokolový model LTE, RLC, MAC, fyzickou vrstvu a rozhraní X2, dále pak aplikace jako jsou Voice-over-IP (VoIP), Constant Bit Rate (CBR). Tento simulační nástroj má i určité limity, podporuje pouze modelování architektury sítě, která je spojena s uživatelskými daty (user plane). To znamená, že je zde možné namodelovat User Equipment, eNodeB, SGW, PGW, PDN. Systémy a bloky, které spadají do kontrolní vrstvy, již namodelovat neumí. Jedná se například o PCRF, OCS. Dále pak podporuje pouze FDD LTE, tedy LTE které pracuje s frekvenčním duplexem. Podpora LTE založeného na TDD zde není implementována. [2][3]

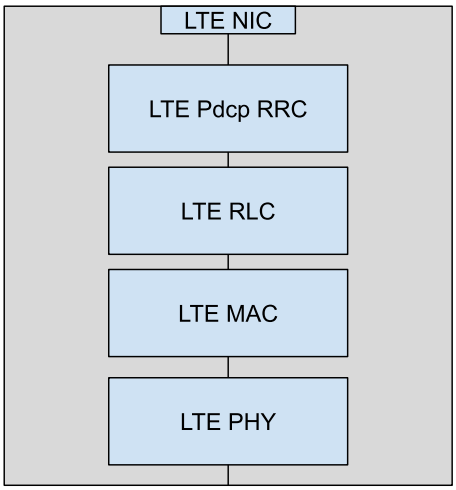
Pro tvoření jednotlivých sítí využívá tento framework tří modulů (obrázek 1.2), tyto mohou být spojovány spolu se sebou samými nebo s ostatními uzly sítě jako je například router, aplikace apod. Jako složené moduly jsou zde implementovány moduly UE a eNodeB. To znamená, že UE a eNodeB jsou dále sestaveny z menších modulů. Každý modul má svůj popisující soubor (.ned), který definuje jeho strukturu a může mít třídní definiční soubor (.cpp, .h), který definuje jeho funkcionality. Třetím modulem je pak speciální modul zvaný Binder, který je viditelný pro všechny ostatní uzly a ukládá si o nich informace. Může být například použit k lokalizaci interferenčních eNodeB, s cílem zjistit mezi-buněčnou interferenci vnímanou User Equipmentem v dané buňce. Binder spravuje také datové struktury obsahující informace o celé síti a může být k němu přistupováno prostřednictvím volání metod každým uzlem v sítí. [1][6]



Obrázek 1.2: *Moduly rádiové přístupové sítě [6]*

UE modul obsahuje dva aplikační moduly (TcpApps, UdpApps), které pro svou funkčnost spoléhají na protokolech transportní vrstvy ISO/OSI referenčního modelu, a to na protokolech TCP a UDP. Tyto dva moduly jsou implementovány jako vektory N nebo M modulů, což umožňuje více aplikací v UE. Jsou postaveny na protokolu IP síťové vrstvy, to vše pak zastřešuje modul LTE NIC (rádiové rozhraní). TcpApps/UdpApps představují jeden konec spojení, další konec může být umístěn v jiném UE nebo někde jinde v topologii. Stejně tak jako TCP a UDP moduly je i modul IP převzat z INET frameworku, v UE modul IP propojuje Network Interface Card (NIC) spolu s aplikacemi, které používají TCP nebo UDP. V eNodeB propojuje tentýž modul také LTE NIC, propojuje ale i modul PPP (Point-to-point Protocol), pomocí kterého se přistupuje do jádra sítě (core network). [1][6]

Moduly LTE NIC v eNodeB a UE reprezentují funkcionalitu čtyř vrstev protokolového modelu LTE, jmenovitě PDCP, RLC, MAC a PHY. Struktura tohoto modulu je zobrazena na obrázku 1.3, je složen ze čtyř podmodulů, které mají vzájemnou korespondenci s protokoly LTE. Je postaven jako rozšíření rozhraní IWirelessNic z frameworku INET, a to proto, aby jej bylo možné snadno zapojit do standardních scénářů. To umožňuje vytvářet scénáře hybridního připojení, např. s uzly vybavenými jak Wi-Fi, tak LTE. Komunikace mezi moduly se provádí pouze prostřednictvím výměny zpráv, takže jakákoliv akce začíná z obsluhy zpráv. Volání mimo moduly jsou používány pouze ve formě getter/setter funkcí, což umožňuje udržovat kontrolu interakcí mezi moduly a tím pádem limitovat možné chybné chování. [1][6]



Obrázek 1.3: *Struktura modulu LTE NIC [6]*

1.3.1 **Modul PdcpRRC**

Modul LTE PdcpRRC propojuje moduly LTE NIC a IP. Modul přijímá data z vyšších vrstev ve směru downlink a ve směru uplink z modulu LTE RLC. Provádí ROHC (Robust Header Compression) a vytváří CID, které spolu s UE ID identifikují spojení v celé síti. Když do tohoto modulu přijde paket, tak je mu přiřazen odpovídající LCID ve formátu <*zdrojová adresa, cílová adresa, zdrojový port, cílový port*>, pokud neexistuje je vytvořeno nové LCID. Následně je paket enkapsulován do Pdcp PDU a odeslán na příslušný RLC port. [1][8]

1.3.2 **Modul RLC**

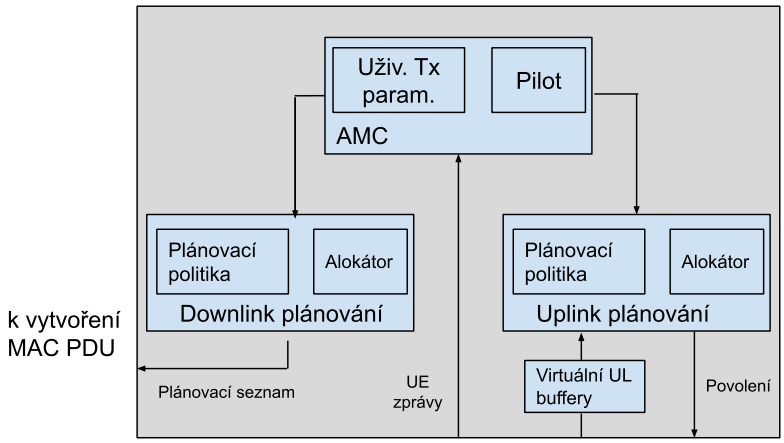
Modul LTE RLC provádí multiplexování a demultiplexování MAC SDU jednotek z nebo do MAC modulu (vrstvy) a implementuje tři RLC módy. Jedná se o transparentní mód (TM), nepotvrzovaný mód (UM) a potvrzovaný mód (AM). Dále přeposílá pakety z PdcpRRC do nebo z příslušných RLC modulů. Modul, který se stará o transparentní mód nemá buffer, protože přeposílá pakety transparentně. Moduly AM a UM však již mají buffery pro přenos a příjem, každý pro jednotlivé CID RLC módu. RLC operace jsou stejné jak v eNodeB, tak v UE. [1]

1.3.3 **Modul MAC**

Většina funkčnosti každého uzlu je umístěna v modulu MAC. Tento modul zajišťuje funkce, jako jsou příjem paketů z vyšších a nižších vrstev, zapouzdření MAC SDU do MAC PDU a naopak, správa zpětné vazby kanálu, adaptivní modulace, kódování a plánování (scheduling). Tyto operace jsou stejné jak pro UE tak pro eNodeB, výjimkou je však plánování a správa zpětné vazby kanálu. Ve směru downlink přicházející MAC SDU z RLC vrstvy jsou uloženy v MAC bufferech, každý pro jedno CID. Na každém TTI (transmission time interval) jsou některé spojení plánovány pro přenos, a to podle odpovídajícího plánovacího seznamu, který vytvoří plánovač. Z plánovaných spojení jsou MAC SDU zapouzdřeny do MAC PDU, které jsou poté uloženy do HARQ bufferů a přeposílány do fyzické vrstvy. Ve směru uplink jsou MAC PDU přicházející z fyzické vrstvy uloženy do HARQ bufferů. Poté je zkontrolována jejich správnost, následně jsou dekapsulovány a přeposílány do RLC vrstvy. [1][7][8]

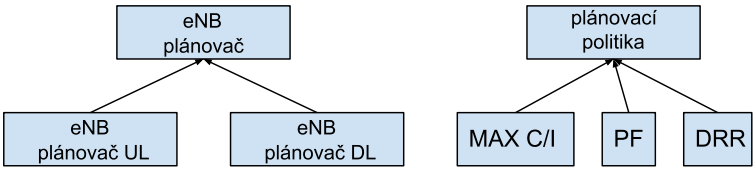
1.3.4 **Přidělování rádiových prostředků**

Přidělování rádiových prostředků se provádí na MAC vrstvě v eNodeB pro uplink i pro downlink. Pro uplink jsou rozhodnutí oznámena UE prostřednictvím tzv. povolovacích zpráv. Každý UE čte tyto zprávy a rozhodne, které připojení bude moci použít pro poskytnuté prostředky. UE žádají o uplink prostředky prostřednictvím Random Access procedury (RAC), která je opět implementována prostřednictvím zpráv generovaných v modulu MAC. Aby bylo možné správně udělat plánovací rozhodnutí, dowlink a uplink plánovače potřebují stav připojení uživatele a identifikátor kvality kanálu každého UE. [1][6]



Obrázek 1.4: *Entity MAC modulu v operaci plánování [1]*

Entita adaptivního modulace a kódování (AMC) ukládá informace o stavu kanálu ze zpráv UE, ukládá také přenosové parametry, jako jsou modulace a kódová rychlost a vypočítává množství bytů na resource blok pro každého uživatele na základě těchto parametrů. Co se týče H-ARQ bufferů (pro přenos a příjem), tak tyto ukládají MAC PDU které jsou posílány, respektive přijímány. To znamená, že MAC PDU jsou v těchto bufferech uloženy, dokud nejsou správně přijaty nebo dokud není dosaženo maximálního počtu opakovaných přenosů. Příjem je informován prostřednictvím zpětné vazby H-ARQ. Vyrovnávací paměti H-ARQ na eNB udržují informace o MAC PDU pro každé připojené UE ve směru downlik i uplink, pro každý proces H-ARQ a pro každé kódové slovo (proto tyto moduly nevyžadují žádnou modifikaci, aby podporovaly Single-User MIMO). ENodeB má tolik bufferů, kolik je k němu připojeno UE v každém směru, kdežto UE má pouze jeden v každém směru, jelikož přímo komunikuje pouze se svým servisním eNodeB. Každý buffer obsahuje N procesů, jeden pro každý H-ARQ proces. Každý proces se skládá ze dvou jednotek, pro podporu SU-MIMO. Hierarchie eNodeB plánovačů je znázorněna na obrázku 1.5. Základní třída plánovače eNB implementuje operace, které jsou společné pro DL a UL, jako je inicializace datových struktur, správa alokace přes alokátor a sběr statistik. Dvě třídy eNodeB plánovače pro UL a DL rozšiřují základní třídu implementací metody *rtxSchedule*, která spravuje opakované přenosy. V závislosti na tom eNodeB plánovač pro UL spravuje požadavky RAC. [1][6][3]



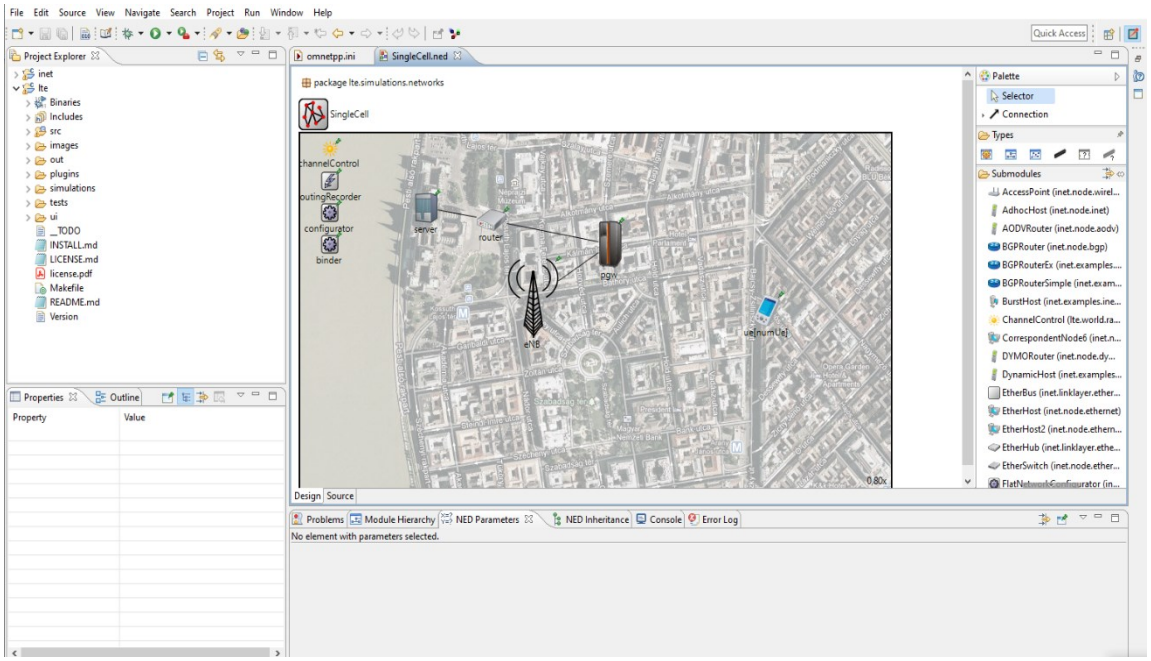
Obrázek 1.5: *Hierarchie eNodeB plánovačů [1]*

Na začátku každého TTI připraví eNB plánovací list v každém směru, který sdílí dostupné zdroje mezi aktivními připojeními podle zvolené politiky. To provádí člen plánovače eNodeB s názvem plánovací politika. Plánovací politika vytváří plánovací listy opakovaným dotazováním alokátoru pro dané množství bajtů přenášených na zkoumaném připojení pomocí hodnoty CQI. To umožňuje předpoklad obecných plánovačů, které rozhodují jak o pořadí priorit připojení, tak o množství dat, která mají být naplánována. Skutečnost, že k plánování dochází ve dvou krocích, umožňuje uživateli spouštět více plánovačů souběžně a vybrat si, který plánovací list se má vybrat mezi sadou alternativ. K implementaci nové plánovací politiky potřebuje vývojář implementovat pouze dva kroky funkce *schedule*. V aktuálním vydání jsou zahrnuty tři velmi známé plánovací politiky, a to Maximum Carrier-over-Interference (Max C/I), Proportional Fair (PF), Deficit Round Robin (DRR). Opakované přenosy lze naplánovat buď s vyšší/nižší prioritou než první přenosy, nebo společně s nimi. [8]

1.3.5 **Modul PHY**

PHY modul implementuje funkce odpovídající fyzické vrstvě LTE protokolového modelu, jako jsou hlášení, přenos a příjem dat a správa kontrolních zpráv. Ukládá fyzické parametry jednotlivých uzlů, jako jsou vysílací výkon a anténní profil (všesměrový nebo anizotropní). To umožňuje definovat makro, mikro a piko eNodeB s rozdílnými vyzařovacími profily. Každý fyzický modul eNodeB a UE má asociován kanálový model – což je C++ třída, která reprezentuje fyzický kanál, jak je vnímán samotným uzlem. Existuje zde tedy rozhraní s názvem *ChannelModel*, které definuje dvě hlavní funkce: *getSINR*, která vrací hodnotu poměru signálu k interferenci a šumu (SINR) a funkci *error*, která kontroluje, zda je paket poškozen. Na straně UE jsou některé úlohy, které souvisejí s funkcemi fyzické vrstvy, prováděny nezávislým modulem *Feedback Generator*. Tento modul generuje CQI, které můžou být konfigurovány jako periodické nebo neperiodické. Fyzické kanály nejsou modelovány až na úroveň OFDM symbolů, a to z důvodu ušetření paměťových prostředků a prostředků procesoru. Jejich funkcionality jsou místo toho implementovány přes kontrolní zprávy, které jsou přenášeny mezi uzly eNodeB a UE. Tyto zprávy jsou posílány přes OMNeT *sendDirect* funkci, která obsahuje ukazatel na odpovídající MAC PDU. Příjemce takové zprávy poté zkontroluje, zda byl přenos úspěšný, a to následujícím způsobem: pro každý resource block *k* obsazený přenosem, je vypočítána hodnota poměru signálu k interferenci a šumu. Tato operace je provedena v *getSINR* funkci, pro každý RB. Toto chování umožňuje simulovat efekty interference pro každý RB, např. vzít v potaz přenos ze sousedních eNodeB. Funkce *error* porovnává CQI použitý pro přenos paketu s tím, který byl vypočítán v okamžiku příjmu. Pravděpodobnost chyby *Perr* se pak získá pomocí realistických BLER křivek a dvou dříve vypočítaných CQI. [6]

1.4 Framework SimuLTE v prostředí OMNeT++



Obrázek 1.6: *Ukázka SimuLTE v prostředí OMNeT++*

V levé horní části prostředí OMNeT++ (obrázek 1.6) se nachází průzkumník projektů. Ten umožňuje uživatelům volit mezi jednotlivými frameworky, projekty a volit jejich obsah. Uprostřed obrázku se nachází NED editor. Na obrázku 1.6 je otevřen jeden z demo příkladů ze SimuLTE, konkrétně *SingleCell*. Mezi textovým a grafickým režimem se dá přepínat pomocí záložek *Design* a *Source* na dolní liště pod editorem. V pravé části se nachází paleta, pomocí které můžeme vytvářet architekturu sítě v grafickém režimu. Umožňuje uživateli využívat nástroje pro tvorbu spojení, elementy, které lze umístit na nejvyšší úroveň našeho NED souboru a poté jednotlivé podmoduly. Simulace se skládají ze souborů *.ned*, *.ini* a *.xml*. [9][10]

1.4.1 NED soubor

Jak už bylo zmíněno výše, NED soubor slouží pro vytvoření architektury sítě. Respektive umožňuje definovat komponenty a sestavit je do větších celků, jako jsou sítě. Umožňuje vytvářet síť pomocí dvou režimů textového a grafického. V grafickém režimu vytváří uživatel síť pomocí grafických ikon. Tyto ikony reprezentují moduly, propojení apod. Propojení jednotlivých modulů je znázorněno pomocí černých linek. Práce v grafickém režimu umožňuje uživateli vytvářet a modifikovat jednotlivé sítě bez předchozí znalosti programování. Pro práci v textovém režimu je potřeba již předchozí znalosti programování. Je založen na NED jazyce a jdou zde nastavit stejné parametry a vlastnosti jako v grafickém režimu. Po vytvoření sítě v textovém režimu, se tato síť přenese i do grafického režimu a naopak. [9][10]

1.4.2 INI soubor

V OMNeT++ jsou simulační modely konfigurovány a jsou jim předávány parametry pro simulaci pomocí INI souboru. Jsou to textové soubory, které lze upravit pomocí libovolného textového editoru. OMNeT++ od verze čtyři však zavádí nástroj navržen pro úpravy souborů INI. Editor souborů INI je součástí IDE OMNeT++ a je velmi efektivní při pomoci uživateli při vytváření popisujících souborů

simulace. Editor má dvě možnosti pro editaci. Konfiguraci lze upravit pomocí formulářů a dialogů nebo jako prostý text. Editor syntaxe navíc podporuje zvýraznění a automatické doplnění. Což je užitečné při vytváření konfiguračních souborů. Mezi jednotlivými režimy se přepíná stejně jako u NED editoru pomocí dolní lišty. Pomocí polí *Form* a *Source. Form* odpovídá režimu pomocí formulářů a *Source* pomocí zdrojového kódu. [9][10]

Pro vygenerování náhodných hodnot v každé simulaci (jako jsou například pozice jednotlivých UE) existuje speciální parametr *seed*. Je nutné zmínit, že se jedná pouze o pseudonáhodný parametr. Pokud je *seed* stejný, vygenerovaná náhodná čísla jsou také stejná. Při každém spuštění simulace budou tedy náhodné hodnoty stejné. Není to chyba, ale důležitá funkce umožňující opakované běhy simulací a obecnou opakovatelnost experimentů. To je výhodné, pokud uživatel sestaví simulaci se zajímavými výsledky a chtěl by ji simulovat opakovaně. Pokud by chtěl uživatel různé náhodné hodnoty proměnných při každém běhu, může například použít jako *seed* ID procesu. Protože při každém startu simulace se s velkou pravděpodobností vygeneruje jiné ID procesu. Ukázka nastavení seedu:

```
seed-set = ${processid}
```

Na začátku simulace je možné nastavit tzv. zahřívací periodu. Je-li tato perioda nastavena, pak výsledky patřící do prvních *x* sekund simulace nebudou započítány do výstupních vektorových a skalárních souborů. To je například užitečné pro simulace v ustáleném stavu.

1.4.3 Grafické běhové prostředí Qtenv

Qtenv je výchozí grafické běhové rozhraní pro simulace. Qtenv podporuje provádění interních simulací, animací, inspekcí, trasování a ladění. Při startu simulace, načte Qtenv INI soubory, které jsou specifikovány v příkazovém řádku a automaticky nastaví simulaci, která je v nich popsána. Pokud jich je zde více, tak se Qtenv zeptá uživatele, kterou simulaci chce spustit. Panel nástrojů v horní části umožňuje přístup k nejčastěji používaným funkcím, jako je například krokování, běh a zastavení simulace. Pod panelem nástrojů se nachází top status bar, což jsou tři pole, která ukazují informace o následující simulační události (může být schován). Pod top status barem se nachází časová osa, která ukazuje události, které nastanou v budoucnu na logaritmické stupnici (může být také schována). Centrální oblast okna je rozdělena do několika dalších oblastí, ve kterých se nacházejí hierarchie objektů, vlastnosti a obsah jednotlivých objektů (obě části se nachází v levé oblasti) a poté síťový displej, který ukazuje danou síť nebo jakýkoliv jiný modul graficky. Pod síťovým displejem se nachází prohlížeč logů. Ten slouží pro výstupy jednotlivých modulů během simulace. [9][10]

1.4.4 Měření statistik v SimuLTE

Záznam výsledků (statistik) simulací lze povolit nebo zakázat na několika úrovních s různými možnostmi konfigurace. Existují zde tři typy statistik: vektorové, skalární a binární. Celkové měření všech statistik nebo pouze uživatelem specifikované statistiky lze zapnout/vypnout pomocí následujícího příkazu:

```
<cesta-k-modulu>.<jméno-statistiky>.statistic-recording = true/false
```

Statistiky jsou ve zdrojovém kódu označeny značkou *@statistic*. Pro vypnutí všech statistik stačí položky *<cesta-k-moulu>* a *<jméno-statistiky>* nahradit znakem *. Stejný princip platí pro už jednotlivé vektorové, skalární i binární statistiky. Níže uvedené příkazy slouží k vypnutí/zapnutí uživatelem definovaných skalárních, vektorových a binárních statistik.

```
<cesta-k-modulu>.<jméno-skaláru>.scalar-recording = true/false
```

```
<cesta-k-modulu>.<jméno-vektoru>.vector-recording = true/false
<cesta-k-modulu>.<jméno-histogramu>.bin-recording = true/false
```

Seznam všech měřitelných statistik, které SimuLTE nabízí, je k dispozici v elektronické příloze D.

1.5 Instalace OMNeT++, INET a SimuLTE

1.5.1 Windows

1.5.1.1 *OMNeT++*

Jako první je potřeba stáhnout archiv OMNeT++ z adresy http://omnetpp.org. Je důležité, aby uživatel umístil stáhnutý archiv do složky, jejíž cesta neobsahuje žádnou mezeru. Dalším krokem je rozbalení archivu. Poté je nutné spustit soubor *mingwenv.cmd.* Dále musí uživatel zadat tyto příkazy:

```
$ ./configure
$ make
```

Dojde k vytvoření binárních souborů pro debug a release. Uživatel by měl poté ověřit instalaci, například spuštěním příkladu *aloha*, a to pomocí těchto příkazů:

```
$ cd samples/aloha
$ ./aloha
```

Uživatel by měl vidět grafická okna a dialogy daného příkladu. Pro zapnutí na eclipsu založeného IDE stačí do *mingwenv (příkazový řádek)* napsat příkaz:

```
$ omnetpp
```

Start z příkazového řádku je doporučován, další možný způsob zapnutí je pomocí *omnetpp.exe* souboru, který se nachází v *ide* adresáři. [24]

1.5.1.2 *INET framework*

Dalším krokem je instalace INET frameworku. Ten je možno nainstalovat při prvním spuštění OMNeT++. Při první návštěvě workbench se totiž OMNeT++ uživatele zeptá, zda chce nainstalovat INET framework. V případě, že uživatel tento krok přeskočí, je zde možnost nainstalovat INET framework v sekci *Help -> Install Simulation Models.* INET framework je také možné nainstalovat přímo ze staženého archivu. Po stažení požadované verze uživatel importuje v OMNeT++ IDE rozbalený archiv pomocí *File -> Import -> Existing Projects to the Workspace.* Poté stačí projekt sestavit pomocí *Project -> Build.* [26]

1.5.1.3 *SimuLTE*

Následně je potřeba importovat SimuLTE framework do nainstalovaného OMNeT++. Aby uživatel mohl spustit poslední verzi SimuLTE (v době psaní této diplomové práce) je potřeba, aby měl OMNeT++ ve verzi 5.6.2 a INET-Framework ve verzi 4.2.2. SimuLTE je možné stáhnout z adresy https://github.com/inet-framework/SimuLTE/releases. Dále je potřeba rozbalit SimuLTE archiv do složky, ve které je uložen rozbalený a importovaný INET framework. Uživatel poté importuje v OMNeT++ IDE rozbalený SimuLTE archiv pomocí *File -> Import -> Existing Projects to the Workspace.* Následně je nutné projekt sestavit (stejně jako v případě INET). [25]

Ve Windows v ukázkových simulacích a příkladech (alespoň co se SimuLTE týče) je pro ukládání výsledků simulace stanovena cesta z určitých proměnných. Viz zdrojový kód níže:

```
output-scalar-file = ${resultdir}/${configname}/${iterationvars}-${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${iterationvars}-${repetition}.vec
```

Proměnná *${iterationvars}* může vkládat do výsledného názvu souboru speciální znaky. Tato proměnná v sobě uchovává hodnoty proměnných parametrů, které existují v simulaci. Pokud v simulaci nějaké proměnné parametry existují, tak proměnná může obsahovat speciální znaky, jako jsou dvojtečka a uvozovky, což samozřejmě Windows jako jméno souboru nedovoluje a simulace nepůjde spustit. Je nutné proto cesty k souborům upravit, například odstraněním této proměnné:

```
output-scalar-file = ${resultdir}/${configname}${repetition}.sca
output-vector-file = ${resultdir}/${configname}${repetition}.vec
```

1.5.2 **Linux**

1.5.2.1 ***OMNeT++***

Aktuálně OMNeT++ nabízí instrukce pro instalaci na Ubuntu 16.04 LTS a Ubuntu 18.04 LTS, Fedora Core 25, Red Hat Enterprise Linux Desktop Workstation 7.x a OpenSUSE 42. Jako první musí uživatel stáhnout z adresy http://omnetpp.org OMNeT++ archiv. Poté je nutné ho rozbalit. OMNeT++ potřebuje, aby jeho složka *bin/* byla v proměnné PATH. Pro dočasné přidání */bin* do PATH stačí vstoupit do OMNeT++ složky a spustit *setenv* skript.

```
$ cd omnetpp-X.X.X
$ . setenv
```

Řetězec *X.X.X* musí uživatel nahradit číselnou verzí OMNeT++, kterou si stáhnul. Pro permanentní nastavení dané proměnné je nutné upravit soubor *.bashrc.* Uživatel pro úpravu může zvolit svůj oblíbený textový editor. Na konec souboru je nutné přidat:

```
export PATH=$HOME/omnetpp-X.X.X/bin:$PATH
```

Poté je nutné zavřít a znovu otevřít terminál pro aplikování změn. Následně v hlavním adresáři OMNeT++ je potřeba spustit skript:

```
$ ./configure
```

Jakmile skript doběhne je možné OMNeT++ zkompilovat. Příkaz pro kompilaci:

```
$ make
```

Uživatel by měl poté ověřit instalaci, například spuštěním příkladu *dyna*, a to pomocí těchto příkazů:

```
$ cd samples/dyna
$ ./dyna
```

Pro start OMNeT++ IDE stačí uživateli do terminálu zadat:

```
$ omnetpp
```

Pro vytvoření ikony na ploše, popřípadě vytvoření ikony v launcheru aplikací slouží tyto dva příkazy:

```
$ make install-menu-item
$ make install-desktop-icon
```

K vytvoření ikony uživatel samozřejmě může použít i linuxové desktopové kontextové menu. [24]

1.5.2.2 ***INET a SimuLTE***

Pro instalaci INET a SimuLTE frameworků lze využít stejného postupu jako pro Windows, který je popsán výše v sekci 1.5.1.2 a 1.5.1.3.

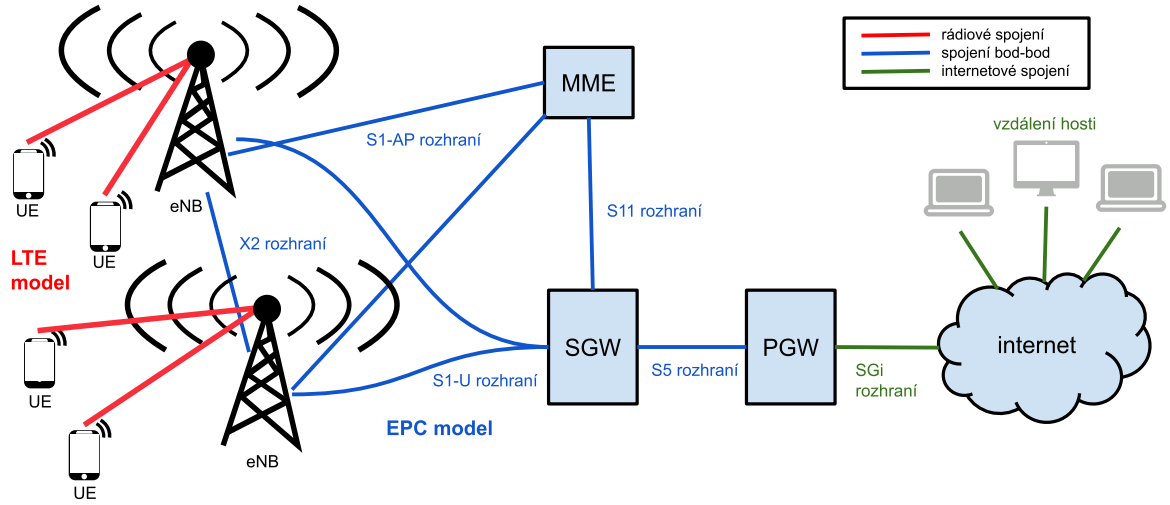
2 Modul LTE pro ns-3

2.1 Ns-3

Network simulator 3 je diskrétní simulátor sítí, ve kterém jsou simulační jádro a moduly implementovány v jazyce C++. Je implementován jako knihovna, která může být staticky nebo dynamicky připojena k hlavnímu C++ programu, který definuje topologii simulace a zahajuje simulaci. Ns-3 taky exportuje veškeré své API do Pythonu, takže umožňuje Python programům importovat ns-3 modul v tom samém smyslu jako u C++. Projekt ns-3 se zavázal k vybudování solidního simulačního jádra, které je dobře zdokumentované, snadno použitelné a laditelné, a které odpovídá potřebám celého simulačního pracovního postupu, od konfigurace simulace po trasování běhu a analýzu. Softwarová infrastruktura ns-3 dále podporuje vývoj simulačních modelů, které jsou dostatečně realistické, aby umožňovaly použití ns-3 jako emulátoru sítě v reálném čase, propojeného s reálným světem a které umožňují opětovné použití mnoha existujících implementací protokolů v reálném světě. [11][12]

2.2 Modul LTE

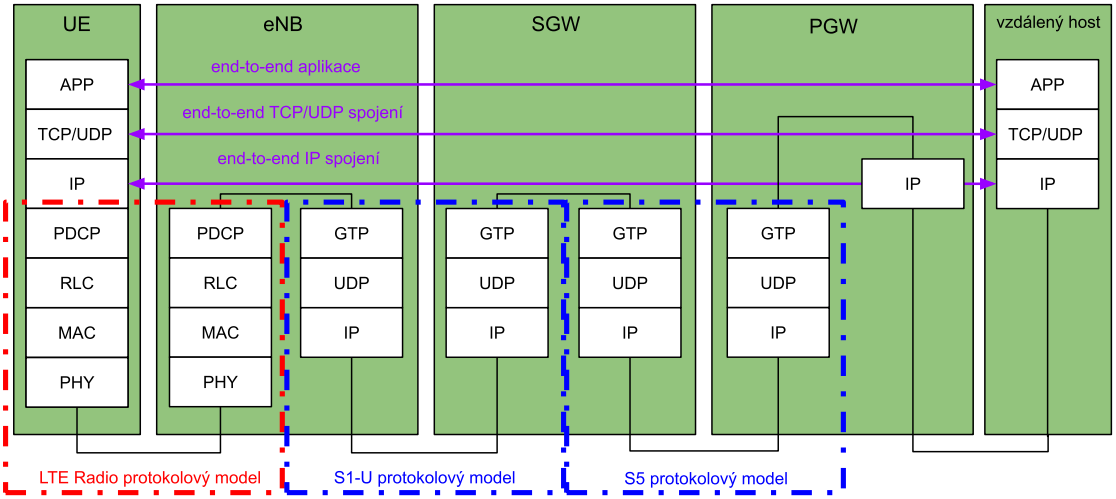
V ns-3 je modul LTE, tedy EPS systém modelován pomocí LTE-EPC network simulátor modelu (LENA) (obrázek 2.1). Jsou zde dvě části, první je LTE model, který obsahuje LTE rádio protokolový model. Jedná se o entity RRC, PDCP, RLC, MAC, PHY. Tyto entity jsou součástí UE a eNodeB uzlů. Druhá část je EPC model, obsahuje rozhraní jádra sítě, protokoly a entity. Tyto entity a protokoly jsou umístěny v uzlech SGW, PGW, MME a částečně v eNodeB uzlech. LTE model je vytvořen tak, aby podporoval evaluaci následujících aspektů LTE systémů: přidělování rádiových prostředků, plánovaní paketů na základě QoS, vnitrobuněčnou koordinaci interference a přístup k dynamickému spektru. Hlavní účel EPC modelu je poskytnout prostředky pro simulaci end-to-end IP konektivity přes LTE. Za tímto účelem je zde podpora pro propojení více UE k internetu přes rádiovou přístupovou síť složenou z více eNB připojených k jádru sítě (obrázek 2.1). [13]



Obrázek 2.1: *Simulační model LTE-EPC* [14]

2.2.1 **Architektura**

Architektura LTE modulu pro ns-3 se skládá z více architektur jednotlivých prvků modulu. Tyto prvky se nazývají modely. V LTE modulu existují architektury pro UE a eNodeB. Tyto architektury obou uzlů se skládají ze dvou částí: architektura modelu LTE rádiového protokolu a architektura modelu PHY kanálu. Dále v EPC modelu existují také dvě architektury, a to pro EPC data plane a EPC control plane. Na obrázku 2.2 je znázorněna architektura LTE-EPC data plane, obrázek ukazuje, jak je namodelována v simulátoru. Obrázek znázorňuje veškeré uzly v datové cestě (tzn. UE, eNodeB, SGW, PGW a vzdáleného hosta v internetu) a všechny protokolové modely specifikované v 3GPP. [15]



Obrázek 2.2*: LTE-EPC architektura data plane [16]*

2.2.2 **Kanál a šíření**

Pro možnosti modelování kanálu používá LTE modul *SpectrumChannel* rozhraní, které je poskytováno modulem *spectrum.* Jsou k dispozici dvě implementace tohoto rozhraní, a to: *SingleModelSpectrumChannel* a *MultiModelSpectrumChannel*. LTE modul vyžaduje pro svou správnou funkci druhé zmíněné rozhraní. To je z důvodu potřeby podpory různých frekvencí a šířek pásma. Všechny modely šíření, které jsou k dispozici v *MultiModelSPectrumChannel* lze použít v LTE modulu. Doporučený model šíření, který se má použít s modulem LTE, je ten, který poskytuje modul *Buildings*, který byl ve skutečnosti navržen speciálně pro LTE. LTE modul bere v úvahu pouze FDD a implementuje downlinkové a uplinkové šíření zvlášť. LTE modul neumožňuje výpočet ztrát šířením mezi: UE a UE, makro základnovou stanicí a makro základnovou stanicí, základnovou stanicí a základnovou stanicí v malých buňkách (piko, femto buňky). [13]

2.2.3 **PHY model**

Tento model odpovídá fyzické vrstvě LTE protokolového modelu, a tedy provádí veškeré funkce spojené s touto vrstvou. Model LTE PHY podporuje modelování antén pomocí třídy *ns-3::AntennaModel*. Proto může být jakýkoli model založený na této třídě přidružen k jakékoli instanci eNodeB nebo UE. Model zahrnuje výpočet mezibuněčného rušení a simulaci uplinkového provozu, včetně paketového přenosu a generování CQI. Model PHY simuluje zpoždění MAC na kanálu v násobcích TTI (1 ms). O tuto část je zpožděn přenos datových i řídících paketů. Co se týče CQI, tak ten se získává měřením SINR a poté předáním tohoto měření modulu *Adaptive Modulation and Coding*, který jej namapuje na index CQI. SINR může být vypočítáno dvěma způsoby v downlink směru a to: *ctrl* metodou nebo *smíšenou* metodou. Jejich rozdíl spočívá v tom, že u *ctrl* metody se berou jako rušící

eNodeB v potaz všechny, i ty které zrovna neprovádí přenos na PDCCH kanálu. Model PHY je založen na Gaussových interferenčních modelech, podle nichž se síly interferujících signálů (v lineárních jednotkách) sečtou dohromady, aby se určila celková interferenční síla. Tento model také podporuje implementaci MIMO, a implementuje model MIMO jako zisk vnímaný přijímačem při použití MIMO s ohledem na zisk získaný při použití SISO. Třídy *UeLtePhy* a *EnbLtePhy* představují fyzickou vrstvu zařízení UE a eNodeB. Dědí ze třídy *LtePhy*, která poskytuje jejich společné funkce. Aby bylo možné samostatně spravovat uplink a downlink, vrstva LTE PHY byla koncipována jako kontejner dvou různých subsystémů základního pásma. PHY model je také zodpovědný za přenos a příjem kontrolních zpráv. [14][17]

2.2.4 **MAC model**

MAC model se v LTE modulu skládá z několika modelů. Skládá se z modelu pro přidělování rádiových prostředků, modelu adaptivní modulace a kódování, modelu transportního bloku, rozhraní *Femto Forum* MAC plánovače, pod toto rozhraní spadá velké množství modulů plánovačů (RR, PF, MT, TTA, BAT, TBFQ, PS, CQA) a z náhodného přístupu. Model pro přidělování rádiových prostředků má na starosti generování specifických struktur zvaných Data Control Indication (DCI), které jsou poté přenášeny pomocí PHY eNodeB do připojených UE, aby je informoval o alokaci zdrojů na základě jednotlivých subrámců. Má na starosti vyplnit některá pole struktury DCI informacemi, jako například: schéma modulace a kódování (MCS), velikost MAC transportního bloku a bitmapu přidělení, která určuje, které RB budou obsahovat data přenášená eNodeB každému uživateli. Pro mapování resource bloků je zde použito lokalizované mapování, kdy v daném subrámci je každý RB alokován stejnému uživateli v obou slotech. Simulátor poskytuje dva modely adaptivní modulace a kódování: jeden založený na již existujícím modelu a druhý založený na modelu fyzické chyby. Druhý zmíněný model je modifikace prvního modelu. Model transportního bloku je zjednodušen s ohledem na specifikace 3GPP. Zejména pro agregaci MAC SDU se používá specifická třída pro simulátor, aby se dosáhlo ekvivalentu simulátoru TB, bez odpovídající složitosti implementace. Rozhraní plánovačů je implementováno jako množina C++ abstraktních tříd. Kdy jednotlivé typy plánovačů (zmíněných výše) jsou implementovány jako tyto třídy. Programátor si poté může vybrat, který typ plánovače zvolí do svého návrhu LTE systému. Modul LTE zahrnuje model náhodného přístupu založený na některých zjednodušených předpokladech. Tento model se skládá z hlavičky náhodného přístupu, odpovědi náhodného přístupu, zpráv a rozlišení obsahu. [13]

2.2.5 **RLC model**

Entita RLC je specifikována v technické specifikaci 3GPP a zahrnuje tři různé typy RLC: transparentní režim (TM), nepotvrzovaný režim (UM) a potvrzovaný režim (AM). Simulátor obsahuje jeden model pro každou z těchto entit. Entity RLC poskytují rozhraní služby RLC do horní vrstvy PDCP a rozhraní služby MAC do spodní vrstvy MAC. Entity RLC používají rozhraní služby PDCP z horní vrstvy PDCP a rozhraní služby MAC ze spodní vrstvy MAC. Implementace potvrzovaného režimu AM RLC se skládá ze tří bufferů. Přenosový buffer, buffer přenesených PDU a buffer pro retransmisi (opakované odeslání). U nepotvrzovaného režimu jsou přenosové operace stejné jako u potvrzovaného režimu, akorát s tím rozdílem, že zde nedochází k znovu odesílání a nejsou zde PDU pro status. U transportního režimu nedochází k přidání RLC hlavičky oproti předchozím režimům, a tím pádem velikost bufferu, která je předána MAC vrstvě je vypočtena pouze součtem velikosti všech paketů připravených pro přenos v přenosovém bufferu. Kromě implementací AM, UM a TM, které jsou modelovány podle specifikací

3GPP, je k dispozici zjednodušený model RLC, který se nazývá saturační režim (SM). Saturační režim RLC simuluje podmínky nasycení, tj. předpokládá, že vyrovnávací paměť (buffer) RLC je vždy plná a může generovat novou PDU, kdykoli je to oznámeno plánovačem. Saturační režim se používá pro zjednodušené simulační scénáře, ve kterých se používá pouze model LTE Radio, bez EPC, a tedy bez jakékoli podpory IP sítí. I když je saturační režim nereálným modelem provozu, stále umožňuje správnou simulaci scénářů s více toky patřícími do různých tříd QoS. [15]

2.2.6 **PDCP model**

PDCP model v simulátoru podporuje funkce, jako jsou: přenos dat – jak pro data plane, tak pro control plane, údržba PDCP SN a přenos SN statusu. Všechny ostatní funkce specifikovány pro PDCP nejsou ještě v simulátoru implementovány. Model je rozdělen na dvě části, *PdcpSapProvider* je poskytován PDCP modelem a používán vyšší vrstvou a *PdcpSapUser* je poskytován vyšší vrstvou a používán PDCP modelem. Obsahuje také dva primitivy, které se starají o přenos a odesílání PDU do vyšší nebo do nižší vrstvy. [18]

2.2.7 **RRC model**

Tento model je přítomen pouze v architektuře LTE-EPC control plane. RRC model poskytuje následující funkcionality. Generování (na eNodeB) a interpretace (na UE) systémových informací, počáteční výběr buněk, funkce navázaní spojení, funkce rekonfigurace, RRC obnovení spojení + handover. Pro popis generování a interpretace existují dva stavové automaty. Počáteční výběr buňky je procedura v nečinném režimu, kterou provádí UE, pokud se nepřipojil k eNodeB. Cílem této procedury je najít vhodnou buňku a připojit se k ní, a tedy získat přístup k celulární síti. Ve většině případů probíhá na začátku simulace. Cílem vyhledávání je tedy detekovat okolní buňky a měřit sílu přijatého signálu z každé z těchto buněk. Měření jsou založena na přijatém RSRP. Použitím měřeného RSRP je model PHY schopen generovat seznam detekovaných buněk, z nichž má každá je označena ID a zprůměrovanou hodnotou RSRP. Tento seznam je pravidelně odesílán do modelu RRC jako protokol o měření. Následuje proces vyhodnocení, kdy model RRC s nasbíraných RSRP vybere buňku s nejlepšími možnými parametry. Tento model také podporuje měření na straně UE. UE dostává periodicky z PHY modulu měření RSRP i RSRQ. Je zde podporováno E-UTRA intra frekvenční měření a agregace nosných frekvencí. Model je založen na konceptu spotřebitele měření UE, což je entita, která může požadovat, aby entita eNodeB RRC poskytovala zprávy o měření UE. Spotřebitelé jsou například handover algoritmus, který počítá rozhodnutí o předání na základě zpráv o měření UE. Spotřebiteli se mohou stát také testovací případy a uživatelské programy. UE prochází přes seznam aktivních měření a kontroluje, zda je splněna podmínka spuštění hlášení měření. Pokud je splněna alespoň jedna spouštěcí podmínka ze všech aktivních konfigurací měření, zahájí se metoda pro hlášení výsledků měření, která vyšším vrstvám ohlásí výsledky měření. [13]

Handover se v simulátoru dělí na dva typy, asistovaný mobilní stanicí a asistovaný sítí. U handoveru asistovaného mobilní stanicí poskytuje UE vstup do sítě ve formě protokolů o měření. Při handoveru asistovaného sítí rozhoduje síť, kdy se má předání spojení spustit a dohlíží na jeho provedení. Simulátor pro funkci handoveru, implementuje algoritmus, jehož hlavním jádrem jsou tři metody. Metoda *AddUeMeasReportConfigForHandover* je používána pro vyžádání zpráv o měření od entity eNodeB předáním požadované konfigurace hlášení. Konfigurace bude použita pro všechna budoucí připojená UE. Metoda *ReportUeMeas* na základě měření nakonfigurovaných z přechozí metody může potvrdit měření eNodeB. Tyto měření jsou poté předány poslední metodě. Metoda *TriggerHandover* rozhodne

na základě daných měření, zda deklaruje handover nebo ne. Kvůli tomu, že některé algoritmy pro opětovné použití frekvencí potřebují mechanismus ovládání výkonu, tak tato funkce byla také implementována v RRC modelu. Pro opětovné použití frekvencí je v LTE modulu implementováno celkem 7 algoritmů. [15][19]

2.3 Ns-3, modul LTE ve vývojovém prostředí a NetAnim

Jelikož tvorba simulačních scénářů v LTE modulu probíhá v programovacích jazycích C++ a python, může si vývojář tvorbu přizpůsobit. Existuje spousta nástrojů pro tvorbu C++ a python kódů, já jsem zvolil prostředí Visual Studio Code. Mezi nejpoužívanější nástroje patří Visual Studio Code, Eclipse a Netbeans. Animace je také důležitým nástrojem pro simulaci sítě. Ns-3 zatím neobsahuje výchozí nástroj pro grafickou animaci, ale v současné době existují dva způsoby, jak výstup simulace animovat, a to pomocí PyViz nebo pomocí NetAnim. PyViz je simulační vizualizér, to znamená, že nepoužívá žádné trasovací soubory. Může být velice užitečný pro účely ladění, tj. dokáže zjistit, zda jsou modely takové, jaké uživatel očekává, kde se pakety nacházejí apod. K dispozici je také integrovaná interaktivní konzole pythonu, kterou lze použít k vlastnímu ladění stavu spuštěných objektů. Ačkoliv je napsán v pythonu, funguje jak pro python, tak pro C++ simulace. Velikou nevýhodou jsou některé známé problémy, jako například ten, že zařízení LTE dosud plně nepodporují tento vizualizér, proto jsem přistoupil ke zvolení druhého vizualizéru, který se jmenuje NetAnim. NetAnim je samostatný spustitelný software založený na Qt4, který používá trasovací soubor vygenerovaný během simulace ns-3 k zobrazení topologie a animaci toku paketů mezi uzly. Vygenerovaný soubor je formátu XML. Kromě toho, poskytuje NetAnim také užitečné funkce, jako jsou tabulky pro zobrazení metadat paketů, zobrazení trajektorie pohybu mobilních uzlů, možnost zobrazit směrovací tabulky různých uzlů v různých časech, možnost zobrazit časovou osu přenosu paketů. Za vytvoření trasovacího XML souboru je zodpovědná třída *ns3::AnimationInterface*. Tato třída používá trasovací infrastrukturu pro trasování cest paketů mezi uzly. Třída zaregistruje dvě události před začátkem simulace, jednu pro příjem a druhou pro vysílání. Pokud je paket připraven pro odeslání nebo příjem, pak jsou odpovídající události zavolány. Když jsou zavolány, třída pak ví, mezi jakými koncovými body bude paket procházet, a tuto informaci přidá do XML souboru, spolu s časovými značkami. [20][21][22]

Spuštění jednotlivých simulačních scénářů lze však také uskutečnit z příkazového řádku linuxu, a to příkazem:

`./waf --run "<název-scénaře> --<parametr>=<hodnota>"`.

Atribut *<název-scénáře>* je jméno souboru, ve kterém se nachází kód dané simulace. Je nutné zmínit, že soubor simulace se musí nacházet v podadresáři */scratch*. Následují parametry z příkazové řádky, kterých může být neomezený počet a v kódu simulace se přidávají pomocí metody *AddValue* třídy *CommandLine.* Každý *<parametr>* by měl mít stejný název jako hodnota proměnné ve zdrojovém kódu (kvůli lepší čitelnosti). Atribut *<hodnota>* je hodnota daného parametru. Níže je uveden příklad zapnutí simulace s názvem *handover* a nastavení parametru *simTime* (čas simulace) na hodnotu 20 sekund:

`./waf --run "handover --simTime=20"`.

Simulace ns-3 mohou být konfigurovány tak, aby umožňovaly produkovat deterministické nebo náhodné výsledky, k tomuto chování je určen tzv. *seed*. Pokud je simulace nakonfigurována s deterministickým *seedem* se stejným číslem běhu, měla by pokaždé vrátit stejné výsledky. Ve výchozím nastavení používá simulace deterministické chování. Třída *ns3::RngSeedManager* poskytuje

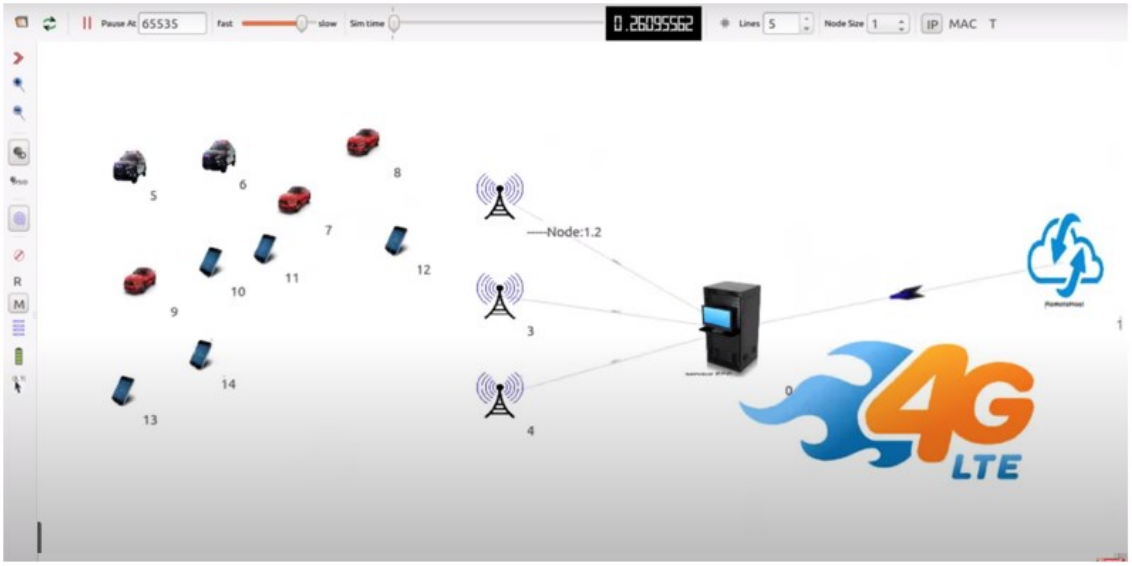
API pro kontrolu náhodných běhů simulace. Tato třída poskytuje dvě metody *SetSeed* – pro nastavení seedu a *SetRun* – pro nastavení čísla běhu, obě metody požadují jako vstup celočíselnou hodnotu. Je nutné říci, že tyto metody musí být zavolány předtím, než dojde k vytvoření jakékoliv náhodné proměnné. [27] Příklad použití třídy *ns3::RngSeedManager*:

```
RngSeedManager::SetSeed(3); // Změna seedu z výchozího 1 na 3
RngSeedManager::SetRun(7); // Změna čísla běhu z výchozího 1 na 7
```

Ovšem není nutné kontrolovat resp. nastavovat náhodnost přímo ve zdrojovém kódu simulace. Uživatel může nastavit náhodný běh simulace taky přímo jako parametr příkazového řádku, pomocí nastavení globální proměnné k tomuto účelu určené. [27]
Příklad spuštění simulace *handover* s číslem běhu 3:

```
$ ./waf --command-template="%s --RngRun=3" --run "handover"
```

2.3.1 NetAnim



Obrázek 2.3: *Ukázka prostředí NetAnim [23]*

Prostředí NetAnim se skládá z horní nástrojové a levé boční nástrojové lišty. Na levé nástrojové listě najdeme různé funkce pro manipulaci s prostředím a zobrazení informací o jednotlivých uzlech. Na horní liště se vybírá XML soubor, který se má zobrazit, poté čas, rychlost simulace a různé úpravy zobrazení simulačního scénáře (jako například zobrazení mřížky na pozadí, zobrazení IP adres, zobrazení chodu paketů v průběhu simulace apod.). Na obrázku 2.3 je simulační prostředí znázorněno i se simulačním scénářem LTE sítě. Uživatel si v kódu v ns-3 simulátoru může nastavit pomocí dříve zmíněného *ns3::AnimationInterface* polohy jednotlivých uzlů, sledování jejich pohybu, a také obrázky jednotlivých prvků sítě. Standardně jsou uzly zobrazovány jako červené body. Je zde možno nastavit i pozadí celé simulace.

NetAnim požaduje, aby každý uzel sítě měl definovanou pozici, jinak nedojde k jeho zobrazení v animaci. Při použití NetAnimu je nutné specifikovat, do jakého výstupního souboru bude animace uložena. Zdrojový kód níže ukazuje možný způsob, jak nastavit jednotlivým uzlům sítě obrázek a také nastavení pozadí celé simulace:

```
AnimationInterface anim ("<cesta-k-souboru>.xml"); // specifikace výstupního souboru
for (uint32_t n = 0; n < numberOfNodes; n++ ) {
```

```
        anim.UpdateNodeImage(nodes.Get(n)->GetId(),
        anim.AddResource("<cesta-k-obrázku-pro-uzel>")); // nastavení obrázku uzlu
        anim.UpdateNodeSize(Nodes.Get(n)->GetId(), 200, 200); // nastavení velikosti
obrázku }
anim.SetBackgroundImage("<cesta-k-obrázku-pro-pozadí>",100,000,2,2,1.0); // nastavení
pozadí a jeho velikosti
```

2.3.2 Měření statistik v ns-3 LTE modulu

Celým smyslem simulace je vygenerovat výsledek pro další zkoumání/zájem. Pro tento problém existuje v ns-3 systém trasování. Trasovací systém umožňuje uživateli sledovat trasovací zdroje definované vývojáři daného modulu. Trasovací systém ns-3 je postaven na konceptech nezávislých trasovacích zdrojů a jednotného připojení zdrojů k požadovaným cílům. Trasovací zdroj může být například přenosová rychlost, zpoždění apod. V ns-3 se pro vybrání daného trasovacího zdroje používá konfigurační subsystém, který k identifikaci trasovacího zdroje používá konfigurační cestu. Konfigurační cesta představuje řetězec ukazatelů objektu. Každý segment cesty odpovídá atributu objektu. Posledním segmentem je požadovaný atribut zájmu. Konfigurační funkce na tento atribut zájmu zavolají příslušnou metodu pro zahájení tvorby statistik. Pro nalezení uživatelem požadované cesty pro daný atribut, který chce sledovat, slouží ns-3 API dokumentace. Na konci každé třídy (například eNodeB) v ns-3 API dokumentaci je seznam všech atributů, seznam všech konfiguračních cest a seznam všech trasovacích zdrojů, které daná třída poskytuje. Příklad konfigurační cesty:

```
/NodeList/*/DeviceList/*/LteEnbRrc/ConnectionEstablished
```

Tato konfigurační cesta slouží k připojení zpětného volání pro trasovací zdroj *ConnectionEstablished*, který se vyvolá vždy, když dojde k připojení UE k eNodeB. K tomuto trasovacímu zdroji je potřeba vytvořit metodu zpětného volání, která musí mít stejnou signaturu jako v dokumentaci. K této signatuře musí být přidán ještě jeden parametr na začátek, a to *std::string context*. Připojení trasovacího zdroje ke konfiguračnímu subsystému poté probíhá pomocí metody *Connect.* Zdrojový kód níže je ukázka vytvoření metody zpětného volání pro trasovací zdroj *ConnectionEstablished* a připojení trasovacího zdroje do konfiguračního subsystému:

```
//metoda zpětného volání
void NotifyConnectionEstablishedEnb ( std::string context, uint64_t imsi,
                                      uint16_t cellid, uint16_t rnti )
{
  std::cout << context << " eNB CellId " << cellid
            << ": successful connection of UE with IMSI " << imsi
            << " RNTI " << rnti << std::endl;
}
// připojení trasovacího zdroje
Config::Connect ("/NodeList/*/DeviceList/*/LteEnbRrc/ConnectionEstablished",
                 MakeCallback (&NotifyConnectionEstablishedEnb));
```

V elektronické příloze I je vypsán seznam všech trasovacích zdrojů v modulu LTE.

2.4 Instalace ns-3 a modulu LTE

V současnosti ns-3 plně podporuje pouze operační systém Linux. Jediné možnosti, jak nainstalovat ns-3 na Windows je pomocí Windows subsystému pro Linux nebo pomocí virtuálního stroje, na kterém Linux poběží. Následná podkapitola se tedy zabývá instalací ns-3 na systému Ubuntu 18.04. [28]

2.4.1 **Linux**

Nejdříve je nutné, aby si uživatel nainstaloval potřebné balíčky pro běh ns-3. Pro C++ uživatele bude potřeba těchto balíčků:

```
$ apt install g++ python3
```

Pro python uživatele bude potřeba těchto balíčků:

```
$ apt install g++ python3 python3-dev pkg-config sqlite3
```

Pro instalaci potřebných balíčků pro běh Netanimu, bude potřeba použít následující příkazy:

```
$ apt install qtbase5-dev qtchooser qt5-qmake qtbase5-dev-tools
```

Pro potřeby ladění programů, bude potřeba použít následující příkazy:

```
$ apt install gdb valgrind
```

Uživatel si pak musí vybrat verzi ns-3 simulátoru. V době psaní této diplomové práce je nejaktuálnější verze 3.35, ukázka instalace bude tedy s verzí 3.35. Poté je potřeba stáhnout požadovanou verzi ns-3 s tím, že je doporučováno použít prostředí *ns3-allinone*. Jedná se o souhrn skriptů, které se starají o stažení a sestavení různých subsystému ns-3 pro uživatele (včetně modulu LTE). Potřebné soubory lze stáhnout a rozbalit pomocí následujících příkazů:

```
$ cd
$ mkdir tarballs
$ cd tarballs
$ wget http://www.nsnam.org/release/ns-allinone-3.35.tar.bz2
$ tar xjf ns-allinone-3.35.tar.bz2
```

Uživatel by měl ověřit, zda složka ns-allinone-3.35 obsahuje tyto soubor:

```
bake            constants.py      ns-3.35                    README
build.py      netanim-3.108   pybindgen-0.22.0   util.py
```

Při prvním sestavení, by měl uživatel sestavit prostředí *allinone*. Toto prostředí totiž nakonfiguruje projekt v té nejčastěji používané podobě. Ve složce *tarballs*, kterou uživatel vytvořil, je nutné spustit python skript s názvem build.py, a to následujícím příkazem:

```
$ ./build.py
```

Pokud sestavení proběhlo v pořádku, měl by uživatel vidět tento text:

```
Build finished successfully (00:02:37)
Leaving directory './ns-3-dev'
```

Pro následnou konfiguraci ns-3 simulátoru je možné použít nástroj *waf* s požadovanými konfiguračními parametry:

```
./waf -d optimized --enable-examples configure
```

Pro ověření instalace stačí spustit python skript s názvem *test.py*. Po dokončení a ověření instalace si uživatel může nainstalovat vývojové prostředí dle jeho preferencí. Je nutné zmínit, že ns-3 nemá oficiální podporu pro žádné vývojové prostředí. [28]

3 Realizace praktických simulačních příkladů

Cílem této kapitoly je seznámit čtenáře s kompletními návody pro čtyři laboratorní cvičení do odborného předmětu v simulátorech OMNeT++ (framework SimuLTE) a ns-3 (modul LTE). Čtenáři bude popsán kompletní návod pro úspěšné vypracování zadaných laboratorních cvičení. Pro úplné pochopení laboratorních cvičení je předpokládaná předchozí znalost programování a znalost základů LTE sítí.

3.1 SimuLTE - Handover v síti LTE

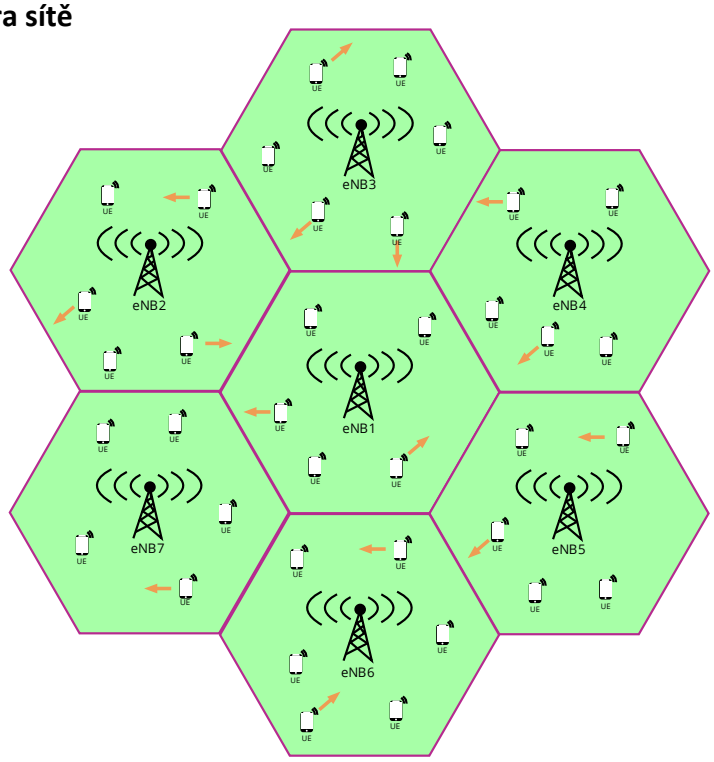
3.1.1 Cíl laboratorního cvičení

Cílem laboratorního cvičení je demonstrace funkcionalit SimuLTE frameworku čtenáři. Vytvořením sítě obsahující prvky architektury LTE, ukázka funkce handoveru a popsání změny určitých parametrů v závislosti na rychlosti UE v LTE síti.

3.1.2 Zadání

Před zpracováním laboratorního cvičení se seznamte s problematikou handoveru v LTE sítích. Vytvořte novou simulaci v simulačním nástroji OMNeT++ a frameworku SimuLTE. Vytvořte nový *.ned* soubor a v něm sestavte architekturu sítě LTE podle obrázku 3.1 v části architektura sítě. V novém souboru *omnetpp.ini* definujte požadovanou funkcionalitu simulace, dle parametrů uvedených v tabulce 3.1. Proveďte simulaci celé LTE sítě s tím, že v každém kroku simulace zvyšujte rychlost UE o 25 km/h se začátkem v 0 km/h a koncem v 200 km/h. Pohyb a pozice UE budou náhodné. Z výstupních statistik simulace si vyberte čtyři vhodné, rozdílné statistiky a jejich výsledky v závislosti na rychlosti UE vykreslete do grafů. Výsledné grafy popište.

3.1.3 Architektura sítě



Obrázek 3.1: *Architektura sítě LTE pro první laboratorní cvičení*

3.1.4 **Parametry simulace**

Tabulka 3.1: *Parametry simulace*

PARAMETR	HODNOTA
Počet resource bloků	25
Šířka pásma	5 MHz
Ztrátový model	ITU-R, Urban Macro
Model mobility	Linear mobility (omnet ++)
Vysílací výkon eNB	40 dBm
Vysílací výkon UE	23 dBm
Šum	5 dB
Rychlost UE	0, 25, 50, 75, 100, 125, 150, 175, 200 km/h
Doba simulace	30 sekund
Počet UE na eNB	10
Počet eNB	7

3.1.5 **Postup řešení**

3.1.5.1 ***Vytvoření nového projektu***

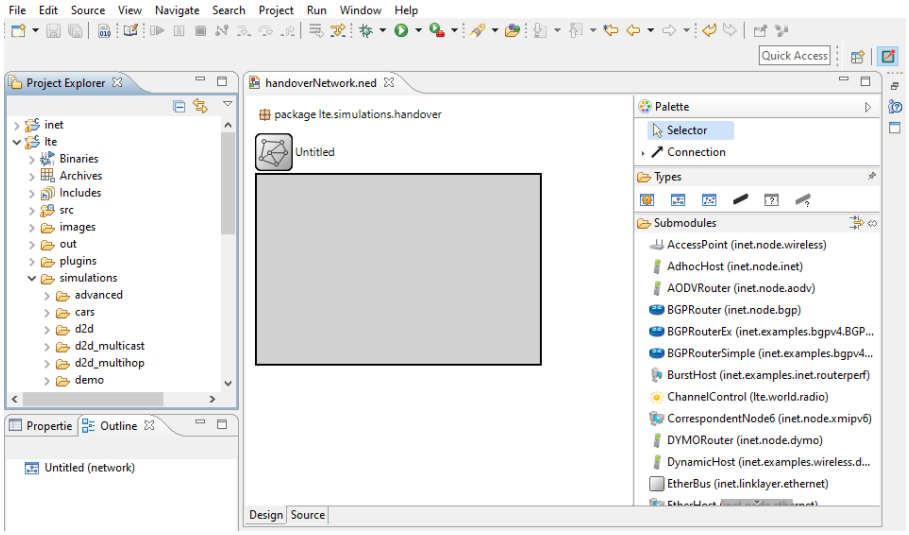
Zapněte simulační nástroj OMNeT++. Ve frameworku SimuLTE (složka lte v project exploreru) ve složce *simulations* vytvořte novou složku s názvem *handover.* V této složce vytvořte tři soubory s názvy *omnetpp.ini*, *ipconf.xml* a *run*. Soubor *omnetpp.ini* slouží pro definování funkcionality sítě, soubor *ipconf.xml* slouží pro nastavení síťových adres všem uzlům simulace a soubor *run* slouží pro samotné spuštění simulace. Soubor *ipconf.xml* bude mít následující obsah:

```
<config>
    <interface hosts='*' address='10.x.x.x' netmask='255.255.255.0'/>
    <!-- <route hosts="*"> -->
</config>
```

Těmito řádky říkáme INET frameworku, že se má použít automatická adresace síťových adres pro všechny uzly sítě s uvedenou síťovou adresou a maskou. V souboru *run* musí být obsaženo:

```
#!/bin/sh
../../src/run_lte $*
```

Ve složce *networks* vytvořte nový *.ned* soubor s názvem *handoverNetwork.ned.* Pomocí *networks -> New -> Network Description File (NED) -> Next -> Ned file with one item*.



Obrázek 3.2: *Prázdný .ned soubor v prostředí OMNeT++*

Výsledkem je soubor pro definici síťové architektury. Vytvořte architekturu sítě podle schématu, a to buď pomocí grafického režimu nebo režimu zdrojového kódu. Mezi grafickým režimem a režimem zdrojového kódu lze přepínat pomocí záložek na spodní liště hlavního okna (záložky *Design* – grafický režim a *Source* – zdrojový kód, viz obrázek 3.2). Architektura sítě se bude skládat ze submodulů: Ue (pole), eNodeB, PgwStandardSimplified, Router, StandardHost, LteBinder, IntegratedCanvasVisualizer, IPv4NetworkConfigurator, RoutingTableRecorder a LteChannelControl. Jednotlivé submoduly lze najít v pravé liště hlavního okna v sekci *submodules.* Submoduly pgw, router, standardhost a jednotlivé eNB propojte pomocí spojení Eth10G. Spojení je k nalezení v pravé liště hlavního okna v sekci *Pallette -> Connection.* Všechny eNB propojte mezi sebou pomocí X2 rozhraní. Jednotlivé pozice všech uzlů sítě lze nastavit již zde, nicméně bude lepší je nastavit až v souboru *omnetpp.ini,* ať jsou všechny parametry simulace na jednom místě. Do konfigurace IPv4NetworkConfigurator přidejte soubor *ipconf.xml* (vytvořený dříve) pomocí příkazu (režim *Source)*:

```
config = xmldoc("ipconf.xml");
```

Ukázková implementace architektury sítě je k dispozici v elektronické příloze A.

3.1.5.2 *Implementace funkcionality sítě*

Pro definování funkcionality sítě, propojení jednotlivých uzlů z hlediska komunikace a nastavení parametrů simulace slouží soubor *omnetpp.ini*. Jako první je potřeba definovat obecné parametry simulace, ty nemají vliv na funkci sítě, nicméně slouží například pro definování cest k obrázkům, nastavení náhodného běhu simulace, doby simulace apod. V souboru *omnetpp.ini* vytvořte konfiguraci s názvem *General.* V konfiguraci definujte jména výstupních souborů pro statistiky – použijte již existujících proměnných prostředí, v případě potřeby nastavte *seed*, vypněte zaznamenávaní všech vektorových i skalárních statistik (potřeba budou pouze některé, ty budou definovány později v konfiguraci), nastavte celkový čas simulace a zahřívací periodu. Lze se inspirovat zdrojovým kódem níže, kde je ukázkové nastavení obecných parametrů pro handover simulaci.

```
################ Obecné parametry #####################
[General]
image-path=../../images
tkenv-plugin-path = ../../../inet/etc/plugins
output-scalar-file-append = false
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec
seed-set = ${repetition}
**.vector-recording = false
**.scalar-recording = false
sim-time-limit = 30s
warmup-period  = 0.3s
```

Další částí konfigurace *General* je nastavení obecných parametrů týkajících se LTE sítě. Jedná se o model šíření, parametry mobility, počet resource bloků a vysílací výkon. V simulaci se bude využívat upravený scénář *URBAN_MACROCELL.* [30] Definujte tyto parametry do konfigurace. Parametry jsou k dispozici v tabulce 3.1. Zapněte slábnutí a stínění signálu, nastavte scénář LTE na *URBAN_MACROCELL,* zapněte uplink a downlink interferenci. Nastavte maximální a minimální omezující oblast na 0 metrů, definujte správný počet resource bloků pro šířku pásma 5 MHz a definujte

vysílací výkon eNodeB a UE. K inspiraci lze použít ukázkové nastavení parametrů v následujícím zdrojovém kódu:

```
################# Model šíření ######################
**.lteNic.channelModel.shadowing = true
**.lteNic.channelModel.fading = true
**.lteNic.channelModel.scenario = "URBAN_MACROCELL"
**.lteNic.channelModel.downlink_interference = true
**.lteNic.channelModel.uplink_interference = true

################ Parametry mobility ####################
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m

############### Počet resource bloků ################
**.numRbDl = 25
**.numRbUl = 25
**.binder.numBands = 25 # tato hodnota by měla být stejná jako počet RBs

############### Vysílací výkon ##################
**.ueTxPower = 23        #dBm
**.eNodeBTxPower = 40    #dBm
```

Dalším krokem je definování parametrů pro správnou funkcionalitu sítě. Vytvořte konfiguraci s názvem *Handover-General.* V této konfiguraci pomocí klíčového slova *network* definujte cestu, kde se nachází *.ned* soubor pro definici síťové architektury. Cesta k souboru v tomto konkrétním případě je následující: *lte.simulations.networks.handoverNetwork.* Pomocí klíčového slova *description* můžete volitelně přidat popis konfigurace. Dalším volitelným krokem je zobrazení adres jednotlivých uzlů sítě a rozhraní pomocí parametru *displayInterfaceTables* - interface table vizualizátoru. Pro příslušné UE nastavte správné obsluhující eNodeB pomocí parametrů *macCellId* a *masterId*, je také potřebné na fyzické vrstvě zapnout handover.

```
**.ue1[*].macCellId = 1
**.ue1[*].masterId = 1

**.ue2[*].macCellId = 2
 …  …  …

# zapnutí handoveru
**.lteNic.phy.enableHandover = true
```

Pro možnosti handoveru je k dispozici v SimuLTE rozhraní X2. Rozhraní X2 je rozhraní, které slouží pro propojení a přenos informací mezi jednotlivými eNodeB, podporuje datovou vrstvu a používá se například pro přepnutí UE ze zdrojového eNodeB k cílovému eNodeB. Pro konfiguraci X2 rozhraní je nutné definovat počet celkových X2 aplikací na každém eNodeB. Počet aplikací je roven počtu připojených eNodeB k danému eNodeB. V SimuLTE pro počet aplikací existuje parametr *numX2Apps.* Je potřeba také správně nastavit jednotlivé porty daných X2 aplikací a pro každou aplikaci přiřadit

adresu připojeného X2 rozhraní ve správném formátu. Formát adresy je následující: *"eNB2%x2ppp0"* - jako první je název a číslo připojeného eNodeB, dále pak znak procenta následován číslem portu rozhraní X2 na připojeném eNodeB. Ve zdrojovém kódu níže, je ukázka správného nastavení připojení X2 rozhraní eNodeB1. Inspirujte se a správně nastavte počet X2 aplikací na všech eNodeB, jednotlivé porty pro dané aplikace a připojení jednotlivých X2 rozhraní všech eNodeB v celé simulaci. Volitelným krokem, avšak doporučeným, je vypnutí zaznamenávání statistik protokolu SCTP, tyto statistky nejsou pro toto laboratorní cvičení důležité, je tedy vhodné je vypnout a omezit tím výslednou velikost výstupních statistik simulace.

```
############### SCTP konfigurace ##################
**.sctp.**.scalar-recording = false   # odstranění nadbytečných SCTP statistik
**.sctp.**.vector-recording = false  # odstranění nadbytečných SCTP statistik
**.sctp.nagleEnabled = false         # pokud je true, přenos malých paketů by byl
zpožděn na X2
**.sctp.enableHeartbeats = false


# X2 konfigurace
*.eNB1.numX2Apps = 6 # jedna x2App na připojený eNodeB


…  …  …


# Nastavení portů serveru (x2App[0]=5000, x2App[1]=5001, ...)
*.eNB*.x2App[*].server.localPort = 5000 + ancestorIndex(1)


# Nastavení připojení X2 rozhraní na eNB1
*.eNB1.x2App[0].client.connectAddress = "eNB2%x2ppp0"
*.eNB1.x2App[1].client.connectAddress = "eNB3%x2ppp0"
*.eNB1.x2App[2].client.connectAddress = "eNB4%x2ppp0"
*.eNB1.x2App[3].client.connectAddress = "eNB5%x2ppp0"
*.eNB1.x2App[4].client.connectAddress = "eNB6%x2ppp0"
*.eNB1.x2App[5].client.connectAddress = "eNB7%x2ppp0"
```

Jak bylo zmíněno dříve, je lepší nastavit pozice jednotlivých uzlů sítě v tomto konfiguračním souboru. Pro nastavení využití pozic definovaných v souboru *omnet.ini* existuje parametr ***.mobility.initFromDisplayString,* ten je potřeba nastavit na hodnotu *false*. Jednotlivé eNodeB jsou rozmístěny do šestiúhelníku se středovým eNodeB s názvem eNB1. Pozici eNodeB lze přiřadit pomocí parametrů *mobility.initialX*, *mobility.initialY* a příslušných hodnotách v metrech. Jednotlivým eNodeB přiřaďte pozice následovně: eNB1 (x = 700m, y = 650m); eNB2 (x = 1100m, y = 100m); eNB3 (x = 1300m, y = 650m); eNB4 (x = 1100m, y = 1250m); eNB5 (x = 300m, y = 1250m); eNB6 (x = 100m, y = 650m); eNB7 (x = 300m, y = 100m). Pozice UE budou podle specifikace zadání náhodné, k tomu slouží funkce *uniform(x_pozice, y_pozice).* Dalším nezbytným parametrem pro nastavení mobility, je vymezení maximální oblasti pohybu jednotlivých UE. Vymezení pohybu UE nastavte na celou oblast simulace tj. 0m < x < 1400m, 0m < y < 1400m. Pro nastavení slouží parametry *constraintAreaMinX* a *constraintAreaMinY.*

```
#ENB POZICE
*.eNB1.mobility.initialX = 700m
```

```
*.eNB1.mobility.initialY = 650m
 …  …  …
#UE pozice
**.ue1[*].mobility.initialX = uniform(600m,800m)
**.ue1[*].mobility.initialY = uniform(550m,750m)
 …  …  …
# VYMEZENÍ POHYBU UE
*.ue1[*].mobility.constraintAreaMinX = 000m
*.ue1[*].mobility.constraintAreaMaxX = 1400m
*.ue1[*].mobility.constraintAreaMinY = 000m
*.ue1[*].mobility.constraintAreaMaxY = 1400m
 …  …  …
```

SimuLTE nabízí velké množství statistik ke sledování v průběhu simulace, které jsou vypsány v elektronické příloze D. Podle zadání protokolu by si měl čtenář vybrat čtyři vhodné, rozdílné statistiky, jejich průběh vykreslit a popsat. Může se také inspirovat tímto návodem, ve kterém jsou vybrány jako čtyři statistiky pro sledování tyto: *cellBlocksUtilization*, *numberOfHandovers*, *harqErrorRate* a *averageCqiDl*. Zvláštní pozornost je potřeba věnovat statistice *numberOfHandovers*, kterou SimuLTE standardně nenabízí a je potřeba si ji ručně přidat do zdrojového kódu frameworku. Pro přidání této statistiky do zdrojového kódu frameworku lze použít následující postup. V souboru *LtePhy.ned*, který se nachází ve složce *lte/src/stack/phy* přidejte k parametrům do třídy *LtePhyUe* nový signál a novou statistiku s názvem *numberOfHandovers.*

```
@signal[numberOfHandovers];
@statistic[numberOfHandovers](title="Number of handovers of the UE";
unit="";source="numberOfHandovers";record=count,vector);
```

Dále v souboru *LtePhyUe.h* ve třídě *LtePhyUe* v chráněné sekci přidejte proměnnou typu *int* s názvem *numberOfHandovers* a *simsignal_t numberOfHandovers_*.

```
/** Number of Handovers */
int numberOfHandovers;
simsignal_t numberOfHandovers_;
```

V souboru *LtePhyUe.cc* ve funkci *initialize* dodejte výchozí hodnotu a daný signál zaregistrujte pod vhodným jménem.

```
numberOfHandovers = 0;
numberOfHandovers_ = registerSignal("numberOfHandovers");
```

Ve stejném souboru ve funkci *doHandover* po dokončení handoveru (funkce *signalHandoverCompleteUe*) inkrementujte proměnnou a emitujte signál *numberOfHandovers_*.

```
void LtePhyUe::doHandover()
{
 …  …  …
ip2lte->signalHandoverCompleteUe();
numberOfHandovers++;
emit(numberOfHandovers_, (long)numberOfHandovers);
 …  …  …
}
```

Tímto způsobem byla přidána nová statistika, která může být sledována v průběhu simulace. Projekt znovu sestavte pomocí kliknutím pravého tlačítka na složku *lte* v *projekt exploreru* a stiskněte možnost *Build project*. Zaznamenávání vybraných statistik lze zapnout (už v souboru *omnetpp.ini*) pomocí následujících příkazů:

```
#Definování výstupních parametrů, které chceme sledovat

**.averageCqiDl:mean.scalar-recording = true
**.averageCqiDl:vector.vector-recording = true

**.cellBlocksUtilizationDl:mean.scalar-recording = true
**.cellBlocksUtilizationUl:mean.scalar-recording = true

**.numberOfHandovers:count.scalar-recording = true
**.numberOfHandovers:vector.vector-recording = true

**.harqErrorRateDl:mean.scalar-recording = true
**.harqErrorRateDl:vector.vector-recording = true
```

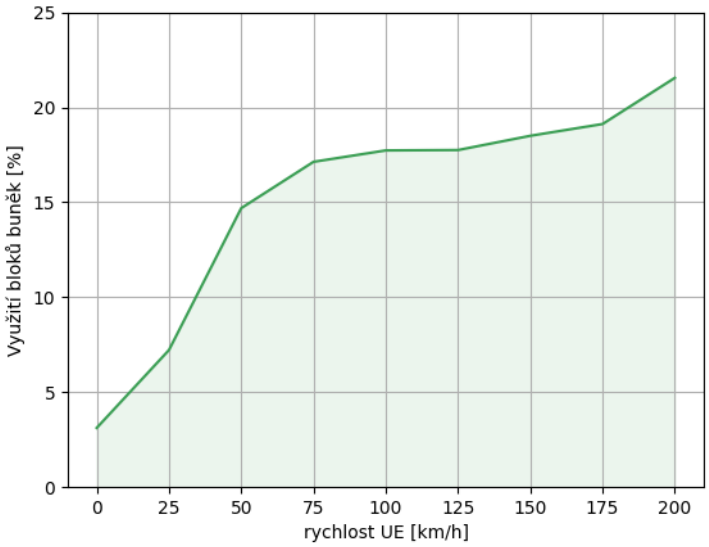
Posledním krokem je definování správné aplikace pro generování provozu v síti. SimuLTE nabízí celkově šest aplikací pro generování provozu: *cbr* – konstantní přenosová rychlost, *burst* – shlukový přenos dat, *d2dMultihop* - aplikace pro šíření upozornění na události pomocí multihop D2D přenosů, *vod* – video na vyžádání (je potřeba mít i vlastní video soubor), *voip* – přenos hlasu přes internetový protokol, *alert* – síření upozornění na události. Pro vytvoření *voip* provozu mezi jednotlivými UE v síti využijte aplikaci *voip*. Vytvořte novou konfiguraci s názvem *VOIP_DL_UP,* která rozšiřuje konfiguraci *Handover-General,* přiřaďte každému UE dvě *udp* aplikace, definujte typ mobility na *LinearMobility* – jedná se o náhodný pohyb (specifikováno v zadání) pomocí parametru *mobilityType.* Dále vytvořte výběrovou proměnnou pro jednotlivé hodnoty rychlosti v tabulce 3.1, rychlosti je nutné převést na metry za sekundu. První UDP aplikaci přiřaďte typ *VoipSender* a druhé *VoipReciever, destinationPort a localPort* bude 3088. Velikost paketu nastavte defaultně na 100 bajtů. K inspiraci použijte zdrojový kód níže:

```
extends = Handover-General
description = Vícero UE komunikuje mezi sebou
**.ue*.numUdpApps = 2
#Jednotlivé rychlosti pohybu UE
**.ue*.mobility.speed = ${speed=0mps,7mps,14mps,21mps,28mps,35mps,42mps,49mps,56mps}
#Nastavení náhodného pohybu UE
**.ue*.mobilityType = "LinearMobility"
#============= Nastavení aplikace VOIP na UE u eNB1=============
*.ue1[*].udpApp[0].typename = "VoIPSender"
 …  …  …
# Definování cílové adresy na UE u eNB1
*.ue1[*].udpApp[0].destAddress = "ue7" + "[" + string(ancestorIndex(1)) + "]"
 …  …  …
*.ue1[*].udpApp[0].destPort = 3088
 …  …  …
*.ue1[*].udpApp[1].typename = "VoIPReceiver"
```

```
 …  …  …
*.ue1[*].udpApp[1].localPort = 3088
 …  …  …
*.ue1[*].udpApp[0].PacketSize =  ${packetSize = 100}
 …  …  …
```

3.1.5.3 *Výstup a zhodnocení výsledků laboratorního cvičení*

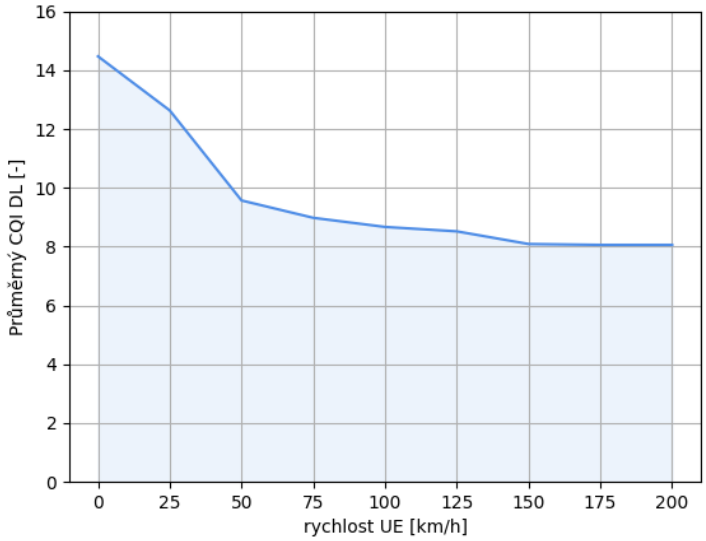
Po dokončení simulace dojde k vytvoření skalárního a vektorového souboru ve složce *results,* která se nachází v adresáři, ve kterém je aktuální simulace. Pomocí těchto souborů lze výsledky simulace vyexportovat a pomocí libovolného nástroje pro vykreslování grafů vykreslit. Pro vykreslení grafů byl použit programovací jazyk Python s knihovnami matplotlib, numpy, scipy a statsmodels. Čtenář může pro vykreslení grafů použít libovolný vykreslovací nástroj dle svého uvážení. Cílem protokolu je výsledné grafy cvičícímu popsat a ověřit tak, že student danou látku správně pochopil. Následují jednotlivé grafy s teoretickým popisem a vysvětlením.



Obrázek 3.3: *Graf využití LTE resource bloků buněk v závislosti na rychlosti UE*

První graf (obrázek 3.3) říká, kolik procentuálně resource bloků je využíváno buňkou v síti (průměrně). V SimuLTE je tento parametr pojmenován jako využití bloků buňky a je považován zejména za důležitý pro řízení QoS provozu LTE. Se zvyšujícím se využitím resource bloků buňky roste pravděpodobnost, že zdroje resp. bloky nemusí být jednotlivým UE přiděleny spolehlivě a včas. To může způsobit zhoršení QoS zejména pro uživatele na okrajích buňky. Z obrázku je patrné, že se se zvyšující se rychlostí UE zvyšuje procentuální využití resource bloků buňky. To je zapříčiněno tím, že jednotlivé buňky provádí daleko větší počet synchronizací, obsluh, vytváření nových a ukončování starých spojení. Všechny handovery v LTE jsou tzv. tvrdé, musí se při každém přepnutí uživatele z jedné buňky do druhé vytvořit nové spojení a ukončit staré. Všechny tyto efekty jsou znásobeny při vyšší rychlosti jednotlivých UE, v síti dochází k daleko většímu počtu handoverů než při nižší rychlosti, a také dochází i k většímu procentuálnímu využití resource bloků buněk. Nicméně v tomto konkrétním

případě se procentuální využití resource bloků dostalo k 23 % při rychlosti 200 km/h, což znamená, že jednotlivé buňky mají pořád dostatek bloků, které mohou přidělit.



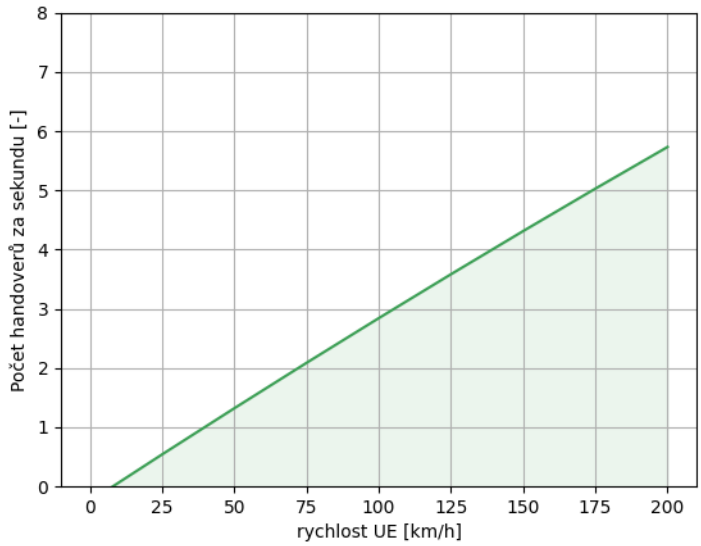
Obrázek 3.4: *Graf závislosti CQI na rychlosti UE*

Druhým grafem je graf identifikátoru popisujícího kvalitu kanálu v závislosti na rychlosti UE (obrázek 3.4). Tabulka CQI ze specifikace ETSI TS 136.213 je k dispozici v tabulce 3.2. [29] V LTE má CQI pouze 16 kódů, od 0 do 15 s tím, že lepším rádiovým podmínkám odpovídá vyšší index CQI. LTE síť na základě informací přijatých z telefonu upravuje kódovací rychlost a modulační schémata. V případě zhoršení rádiových podmínek pak dojde k nastavení nižší přenosové rychlosti. Z grafu je patrné, že s rostoucí rychlostí pohybu jednotlivých UE dochází ke zmenšení identifikátoru kvality kanálu. Největší pokles CQI je mezi rychlostmi 0 až 50 km/h, u dalších rychlostí dochází k rovnoměrnějšímu poklesu. Identifikátor kvality kanálu klesá s rostoucí rychlostí z toho důvodu, že když se UE rychle pohybuje, pohybuje se z oblastí z vyšším pokrytím signálu do oblastí s nižším pokrytím signálu (horší rádiové podmínky), jinými úrovněmi slábnutí signálu, po různých cestách. Čím rychleji se pohybuje, tím častěji na něj působí tyto rozdílné vlastnosti, což má za následek celkovou změnu hodnoty identifikátoru kvality kanálu.

Tabulka 3.2: *CQI tabulka podle specifikace ETSI TS 136.213 [29]*

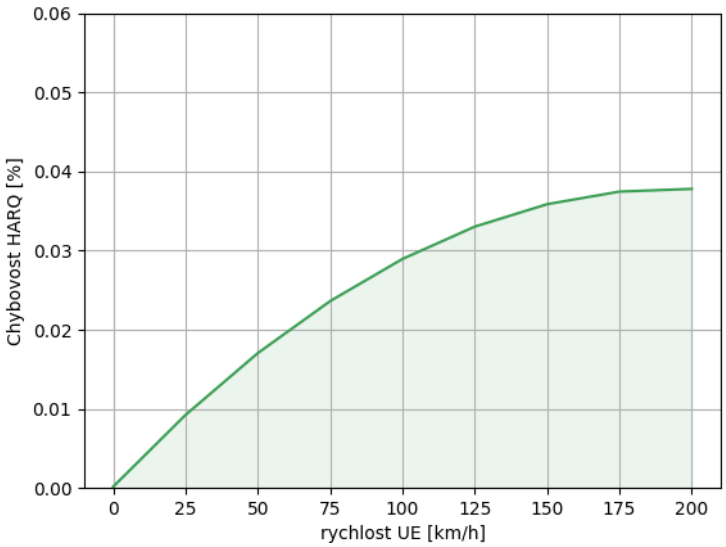
CQI index	modulace	rychlost kódování	efektivita
0	mimo dosah		
1	QPSK	78	0,1523
2	QPSK	193	0,3770
3	QPSK	449	0,8770
4	16QAM	378	1,4766
5	16QAM	490	1,9141
6	16QAM	616	2,4063
7	64QAM	466	2,7305
8	64QAM	567	3,3223
9	64QAM	666	3,9023
10	64QAM	772	4,5234
11	64QAM	873	5,1152
12	256QAM	711	5,5547
13	256QAM	797	6,2266
14	256QAM	885	6,9141
15	256QAM	948	7,4063

Přidáním nové statistiky do zdrojového kódu simulátoru umožníme statistiku sledovat v průběhu simulace, na konci simulace pak simulátor umožňuje statistiku vykreslit. Pokud čtenář nechce vykreslovat statistiky pomocí externích nástrojů, může využít již integrovaného GUI v OMNeT++.



Obrázek 3.5: *Graf závislosti počtu handoverů za sekundu na rychlosti UE*

Z obrázku 3.5 je vidět, že s rostoucí rychlostí UE roste počet handoverů za sekundu v síti, je to z toho důvodu, že čím rychleji se UE pohybuje, tím častěji dochází ke změně základnové stanice. Od určitých vysokých rychlostí může dojít k problému, kdy rychlost pomocí které může architektura sítě předat uživatele z jedné buňky do druhé je pomalejší, než rychlost pohybu uživatele. V takovém případě se pravděpodobně uživatel dostane do a mimo dosah eNodeB rychleji než je síť schopná zpracovat. Když k tomu dojde, je pro síť jednoduše příliš náročné a složité obsloužit uživatele. Uživatel poté nebude mít žádnou službu (nebude připojen k eNodeB) nebo bude mít extrémně nespolehlivý signál, s velice špatnými parametry spojení.



Obrázek 3.6: *Graf závislosti chybovosti HARQ na rychlosti UE*

Obrázek 3.6 popisuje závislost hybrid automatic repeat request (HARQ) chybovosti na rychlosti UE. HARQ proces závisí na přijetí ACK pro pakety. V případě, že odesílatel neobdrží potvrzení (ACK) před vypršením časového limitu, se špatně zaslaný paket nezahodí, ale uloží do vyrovnávací paměti. Základní myšlenkou pak je, že dva nebo více paketů přijatých s nedostatečnými informacemi lze zkombinovat dohromady takovým způsobem, že lze dekódovat celkový signál. Chybovost při tomto procesu se poté označuje jako chybovost HARQ. Z obrázku je patrné, že při zvyšující se rychlosti UE roste chybovost HARQ, nicméně chybovost při rychlosti 200 km/h dosahuje 3,8 %, tudíž je malá i při takto velkých rychlostech. Při vysokých rychlostech UE nemusí být špatný paket opětovně zaslán správně, a to například z toho důvodu, že UE přešel do jiné oblasti s horšími rádiovými podmínkami.

3.2 SimuLTE – Interference ve dvouvrstvé LTE síti

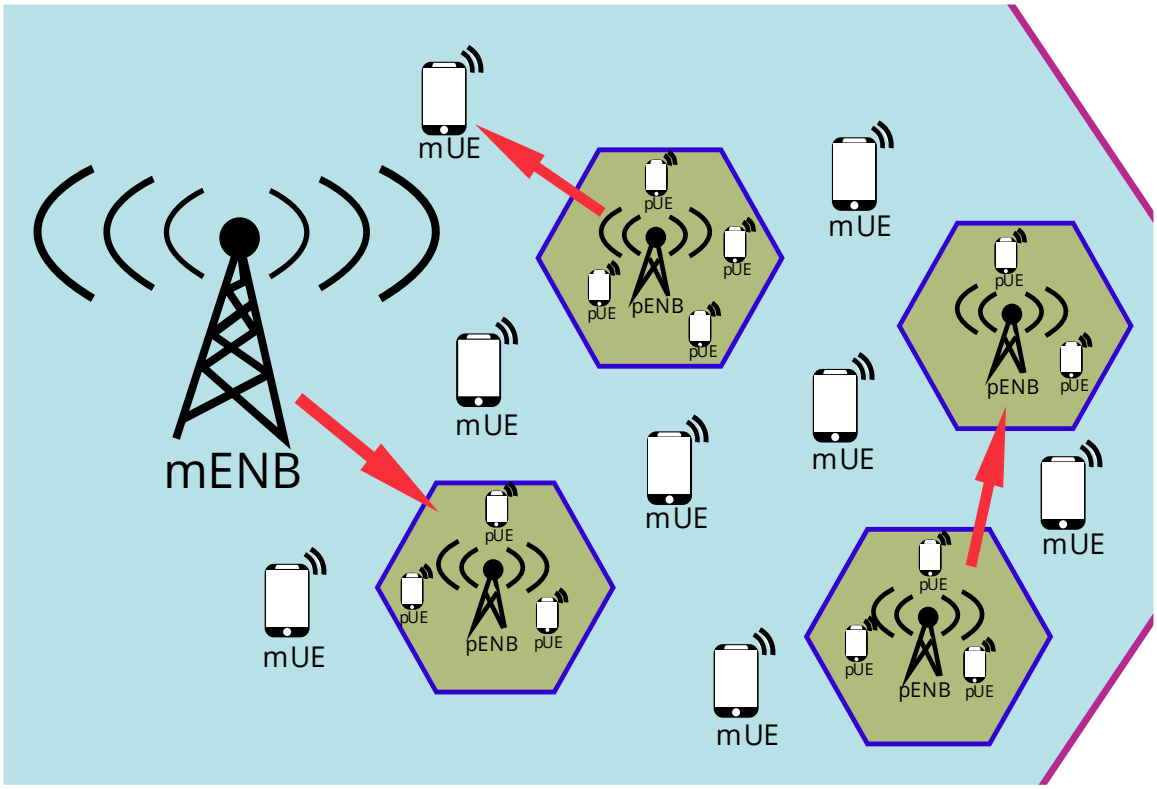
3.2.1 Cíl laboratorního cvičení

Cílem laboratorního cvičení je demonstrace funkcionalit SimuLTE frameworku čtenáři. Vytvoření sítě obsahující prvky architektury LTE, ukázka a pochopení problému interference v síti LTE. Popsání změny určitých parametrů v piko buňkách v závislosti na vysílacím výkonu makro buňky a typu alokace.

3.2.2 Zadání

Před zpracováním laboratorního cvičení se seznamte s problematikou interference v LTE sítích. Vytvořte novou simulaci v simulačním nástroji OMNeT++ a frameworku SimuLTE. Vytvořte nový *.ned* soubor a v něm sestavte architekturu sítě LTE podle obrázku 3.7 v části architektura sítě. V novém souboru *omnetpp.ini* definujte požadovanou funkcionalitu simulace, dle parametrů uvedených v tabulce 3.7. Proveďte simulaci celé LTE sítě s tím, že v každém kroku simulace zvyšujte vysílací výkon makro eNodeB o 2,5 dBm se začátkem v 30 dBm a koncem v 52,5 dBm. Tento postup proveďte pro každý typ alokace, pro náhodou, souvislou a plnou alokaci. Rychlost UE bude nastavena na 1,5 metrů za sekundu. Z výstupních statistik simulace si vyberte čtyři vhodné, rozdílné statistiky a jejich výsledky v závislosti na vysílacím výkonu makro eNodeB vykreslete do grafů. Výsledky pro jednotlivé typy alokací budou vykresleny do stejného grafu. Výsledné grafy popište.

3.2.3 Architektura sítě



Obrázek 3.7: *Architektura sítě LTE pro druhé laboratorní cvičení*

Legenda:

- interference

mUE - makro user equipment

pUE - piko user equipment

mENB - makro eNodeB

pENB - piko eNodeB

3.2.4 **Parametry simulace**

Tabulka 3.3: *Parametry druhé simulace*

PARAMETR	HODNOTA
Počet resource bloků	50
Šířka pásma	10 MHz
Poloměr makro buňky	1000 m
Poloměr piko buňky	100 m
Počet pUEs v piko buňce	4
Počet piko buněk	4
Vysílací výkon makro eNodeB	30 - 52,5 dBm
Vysílací výkon piko eNodeB	27 dBm
Vysílací výkon UE	23 dBm
Prostředí	ITU-R, Urban macrocell
Výška budov	10 m
Čas simulace	30 s

3.2.5 **Postup řešení**

3.2.5.1 ***Vytvoření nového projektu***

Zapněte simulační nástroj OMNeT++. Ve frameworku SimuLTE (složka lte v project exploreru) ve složce *simulations* vytvořte novou složku s názvem *interference.* V této složce vytvořte tři soubory s názvy *omnetpp.ini*, *ipconf.xml* a *run*. Soubor *omnetpp.ini* slouží pro definování funkcionality sítě, soubor *ipconf.xml* slouží pro nastavení síťových adres všem uzlům simulace a soubor *run* slouží pro samotné spuštění simulace. Soubor *ipconf.xml* bude mít následující obsah:

```
<config>
    <interface hosts='*' address='10.x.x.x' netmask='255.255.255.0'/>
    <!-- <route hosts="*"> -->
</config>
```

Těmito řádky říkáme INET frameworku, že se má použít automatická adresace síťových adres pro všechny uzly sítě s uvedenou síťovou adresou a maskou. V souboru *run* musí být obsaženo:

```
#!/bin/sh
../../src/run_lte $*
```

Ve složce *networks* vytvořte nový .ned soubor s názvem *interferenceNetwork.ned.* Pomocí *networks -> New -> Network Description File (NED) -> Next -> Ned file with one item*. Výsledkem je soubor pro definici síťové architektury. Vytvořte architekturu sítě podle schématu a to buď pomocí grafického režimu nebo režimu zdrojového kódu. Architektura sítě se bude skládat ze submodulů: UE, eNodeB, PgwStandardSimplified, Router, StandardHost (server), LteBinder, IntegratedCanvasVisualizer, IPv4NetworkConfigurator, RoutingTableRecorder, LteChannelControl a ExtCell. Submoduly pgw, router, standardhost (server) a jednotlivé eNB propojte pomocí spojení Eth10G. Spojení je k nalezení v pravé liště hlavního okna v sekci *Pallette -> Connection.* Všechny eNB propojte mezi sebou pomocí X2 rozhraní. Externí buňku nepropojujte s žádným uzlem, externí buňka pracuje samostatně bez nutnosti připojení. Do architektury sítě přidejte pouze piko UE, jelikož externí buňka v SimuLTE je zjednodušený eNodeB, který nemá žádné UE, které by obsluhoval, ale vytváří interferenci s UE

umístěnými v síti. Toto je samozřejmě užitečný mechanismus pro simulaci mezibuněčné interference bez potřeby přidávání dalších a dalších eNodeB spolu s jejich obsluhovanými UE. Jedná se tedy o zjednodušení architektury, jelikož není potřeba přidávat k této makro buňce obsluhované UE – o to už se postará framework sám. Přesné pozice jednotlivých uzlů nastavte až v souboru *omnetpp.ini,* ať jsou všechny parametry simulace na jednom místě. Do konfigurace IPv4NetworkConfigurator přidejte soubor *ipconf.xml* (vytvořený dříve) pomocí příkazu (režim *Source)*:

```
config = xmldoc("ipconf.xml");
```

Ukázková implementace architektury sítě je k dispozici v elektronické příloze B.

3.2.5.2 *Implementace funkcionality sítě*

Stejně jako tomu bylo v případě prvního protokolu, je nutné v souboru *omnetpp.ini* definovat funkcionalitu sítě, propojení jednotlivých uzlů apod. Je vhodné si určité části zdrojového kódu rozdělit do jednotlivých konfigurací pro lepší orientaci a pro možnost dalšího využití těchto konfigurací i v jiných konfiguracích. Prvním krokem je definice obecných parametrů simulace. V souboru *omnetpp.ini* vytvořte konfiguraci s názvem *General.* Do této konfigurace nastavte jména výstupních souborů pro statistiky - použijte již existující proměnné, cestu k *.ned* souboru pro architekturu sítě (*lte.simulations.networks.interferenceNetwork*), volitelnými kroky jsou nastavení pozadí a definování *seedu*. Dále vypněte zaznamenávání všech vektorových i skalárních statistik (potřebné statistky budou povoleny v konfiguraci *interference-general*), nastavte celkovou dobu běhu simulace a zahřívací periodu (dle potřeby). Ukázkové nastavení obecných parametrů je k dispozici ve zdrojovém kódu níže.

```
################## Obecné parametry ######################
[General]
image-path=../../images
tkenv-plugin-path = ../../../inet/etc/plugins
output-scalar-file-append = false
output-scalar-file = ${resultdir}/${configname}/${repetition}.sca
output-vector-file = ${resultdir}/${configname}/${repetition}.vec

network = lte.simulations.networks.interferenceNetwork
**.vector-recording = false
**.scalar-recording = false

seed-set = ${repetition}

sim-time-limit = 30s
warmup-period  = 0.3s
```

Další částí konfigurace *General* je nastavení obecných parametrů týkajících se LTE sítě. Jedná se o výšku budov, povolení interference z externí buňky, parametry mobility, počet resource bloků a vysílací výkon. V simulaci se bude využívat upravený scénář *URBAN_MACROCELL.*[30] Definujte tyto parametry do konfigurace. Parametry jsou k dispozici v tabulce 3.3. Nastavte scénář modelu kanálu na *URBAN_MACROCELL*, povolte interferenci z externí buňky pomocí parametru *extCell_interference*, nastavte výšku budov, nastavte maximální a minimální omezující oblast na 0 metrů, definujte správný

počet resource bloků pro šířku pásma 10 MHz a definujte vysílací výkon eNodeB a UE. Vzorové nastavení těchto parametrů je na zdrojovém kódu níže.

```
################ Parametry mobility #####################
**.mobility.constraintAreaMinZ = 0m
**.mobility.constraintAreaMaxZ = 0m
**.lteNic.channelModel.scenario = "URBAN_MACROCELL"
**.lteNic.channelModel.building_height = 10
**.lteNic.channelModel.extCell_interference = true

################ Počet resource bloků ################
**.numRbDl = 50
**.numRbUl = 50
**.binder.numBands = 50  # tato hodnota by měla být stejná jako počet RBs

################ Vysílací výkon ##################
**.ueTxPower = 23       #dBm
**.eNodeBTxPower = 27   #dBm
```

Jako následující krok vytvořte konfiguraci s názvem *Interference-General.* V této konfiguraci můžete volitelně přidat popis konfigurace pomocí klíčového slova *description* a pomocí parametru *interfaceTableVisualizer.displayInterfaceTables* zobrazit IP adresy všech uzlů v síti po zapnutí simulace. Důležitým parametrem pro vypnutí přebírání souřadnic jednotlivých uzlů z *.ned* souboru je parametr mobility *initFromDisplayString.* Při jeho vypnutí bude SimuLTE Framework používat souřadnice ze souboru *omnetpp.ini*. Čímž samozřejmě dojde k lepší přehlednosti, jelikož budou všechny potřebné informace pro simulaci ve stejném souboru. Vypněte přebírání souřadnic z *.ned* souboru, připojte jednotlivé piko UE k jím příslušícím piko eNodeB pomocí parametrů *macCellId* a *masterId*. Pozici eNodeB lze přiřadit pomocí parametrů *mobility.initialX*, *mobility.initialY.* Jednotlivým eNodeB přiřaďte pozice následovně: eNB1 (x = 400m, y = 550m); eNB2 (x = 450m, y = 150m); eNB3 (x = 650m, y = 200m); eNB4 (x = 700m, y = 590m); extCell (x = 250m, y = 250m). Umístěte UE příslušící daným eNodeB do kruhové oblasti s maximálním poloměrem 100 metrů. Středovým bodem budou souřadnice obsluhujícího eNodeB. Pro náhodné rozmístění použijte funkci *uniform.* Pro vymezení pohybu jednotlivých UE nastavte stejné hodnoty jako pro jejich náhodné rozmístění.

```
[Config Interference-General]
**.mobility.initFromDisplayString = false

# Připojení každého UE k eNB
**.ue1*.macCellId = 1
**.ue1*.masterId  = 1
   …   …   …
#eNB POZICE
*.eNB1.mobility.initialX = 400m
*.eNB1.mobility.initialY = 550m
    …   …   …
# pozice makro buňky
*.extCell[0].position_x = 250m
```

```
*.extCell[0].position_y = 250m


#UE pozice
**.ue1*.mobility.initialX = uniform(360m,430m)
**.ue1*.mobility.initialY = uniform(520m,580m)
    …   …   …
# vymezení pohybu UE
*.ue1*.mobility.constraintAreaMinX = 360m
*.ue1*.mobility.constraintAreaMaxX = 430m
*.ue1*.mobility.constraintAreaMinY = 520m
*.ue1*.mobility.constraintAreaMaxY = 580m
    … … …
```

Dále pomocí parametru *numExtCells* přiřaďte správný počet externích buněk simulaci. Definujte dvě výběrové proměnné pro vysílací výkon makro buňky a pro typy alokací. Jednotlivé typy alokací, které framework SimuLTE nabízí, jsou: *FULL_ALLOC* – plná alokace, *RANDOM_ALLOC* – náhodná alokace a *CONTIGUOUS_ALLOC* – souvislá alokace. Nastavte procentuální počet alokovaných resource bloků na 50 %, k jeho nastavení slouží parametr *bandUtilization*. V případě plné alokace alokuje buňka veškeré dostupné resource bloky v LTE rámci. U náhodné alokace alokuje buňka v rámci X resource bloků. Bloky, které budou alokovány, jsou vybírány náhodně a počet resource bloků, které budou alokovány je závislý na parametru *bandUtilization*. V případě posledního typu alokace (souvislá alokace) dochází k alokování souvislého bloku X resource bloků, který začíná od určitého resource bloku. Pro počet alokovaných resource bloků se zde také využívá parametr *bandUtilization.* Pro nastavení požadovaných parametrů a inspiraci lze použít zdrojový kód níže:

```
# Externí buňka
*.numExtCells = 1


# Vysílací výkon makro buňky
*.extCell[*].txPower = ${txPower= 30, 32.5, 35, 37.5, 40, 42.5, 45, 47.5, 50, 52.5}


# Typy alokací
*.extCell[*].bandAllocationType =
${extAllocType="FULL_ALLOC","RANDOM_ALLOC","CONTIGUOUS_ALLOC"}


# Procentuální počet alokovaných resource bloků
*.extCell[*].bandUtilization = 0.5
```

Podle zadání protokolu by si měl čtenář vybrat čtyři vhodné, rozdílné statistiky, jejich průběh v závislosti na vysílacím výkonu makro buňky a typu alokací vykreslit a popsat. Může se také inspirovat tímto návodem, ve kterém jsou vybrány jako 4 statistiky pro sledování tyto: *rcvdSinr*, *averageCQI*, *avgServedBlocks* a *rlcCellThroughput*. SimuLTE nabízí velké množství statistik ke sledování v průběhu simulace, které jsou k dispozici v elektronické příloze D. Zaznamenávání vybraných statistik zapněte (soubor *omnetpp.ini*) pomocí následujících příkazů:

```
#Definování výstupních parametrů, které chceme sledovat
**.um.rlcCellThroughputDl:mean.scalar-recording = true
**.um.rlcCellThroughputDl:vector.vector-recording = true
```

```
**.rlcCellThroughputUl:mean.scalar-recording = true
**.rlcCellThroughputUl:vector.vector-recording = true

**.averageCqiDl:mean.scalar-recording = true
**.averageCqiDl:vector.vector-recording = true

**.averageCqiUl:mean.scalar-recording = true
**.averageCqiUl:vector.vector-recording = true

**.avgServedBlocksDl:mean.scalar-recording = true
**.avgServedBlocksDl:vector.vector-recording = true

**.avgServedBlocksUl:mean.scalar-recording = true
**.avgServedBlocksUl:vector.vector-recording = true

**.rcvdSinr:mean.scalar-recording = true
**.rcvdSinr:vector.vector-recording = true
```

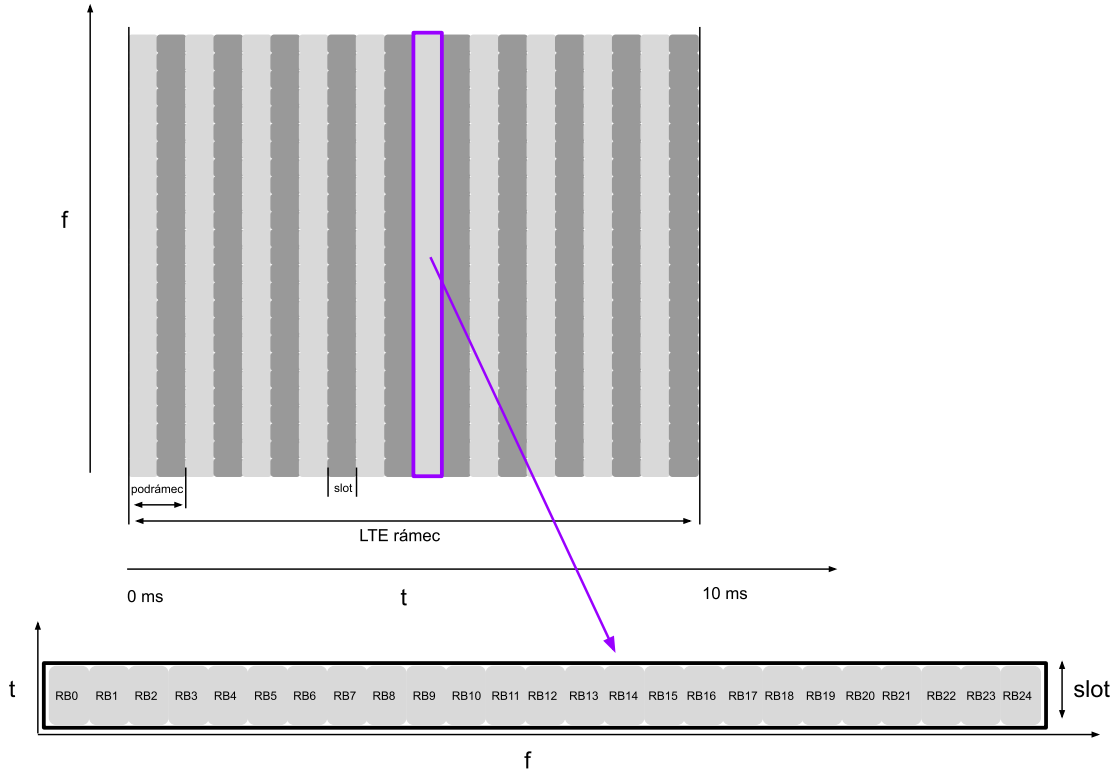
Z pravidla posledním krokem před spuštěním simulace a vygenerováním výsledků je potřeba definování správné aplikace pro generování provozu v síti. Vytvořte konfiguraci *VOIP,* ve které definujete *voip* provoz mezi serverem a UE v síti, pro tento účel využijte SimuLTE aplikaci *voip.* Konfigurace *VOIP* bude rozšiřovat konfiguraci *Interference-General.* Každému UE přiřaďte jednu *udp* aplikaci, definujte typ mobility na *LinearMobility* pomocí parametru *mobilityType.* Nastavte správnou rychlost UE a správně definujte aplikaci *voip*. Serveru přiřaďte správný počet *udp* aplikací pomocí parametru *numUdpApps.* Nastavte typ aplikace na UE jako *VoipReciever, destPort* serveru a *localPort* UE bude 3000, nastavte vhodnou výchozí velikost paketu. Aplikace na serveru bude typu *VoipSender* a čas začátku vysílání *udp* dat bude uniformní mezi 0 a 0,02 sekundami. K inspiraci lze použít zdrojový kód níže:

```
# jedna UDP apliakce pro každého uživatele
*.ue*.numUdpApps = 1
   …  …  …
# rychlost pohybu UE
**.ue*.mobility.speed = 1.5mps

# typ mobility pro UE
**.ue*.mobilityType = "LinearMobility"

#============= Nastavení aplikace VOIP =============
*.ue*.udpApp[*].typename = "VoIPReceiver"
*.ue*.udpApp[0].localPort = 3000
# definování velikosti paketu
*.server.udpApp[*].PacketSize = 50
# Definování cílové adresy na serveru pro UE1
*.server.udpApp[0..3].destAddress = "ue1" + string(ancestorIndex(0)+1)
   …  …   …
```

```
*.server.udpApp[*].destPort = 3000
*.server.udpApp[*].localPort = 3088+ancestorIndex(0)
*.server.udpApp[*].typename = "VoIPSender"
*.server.udpApp[*].startTime = uniform(0s,0.02s)
```

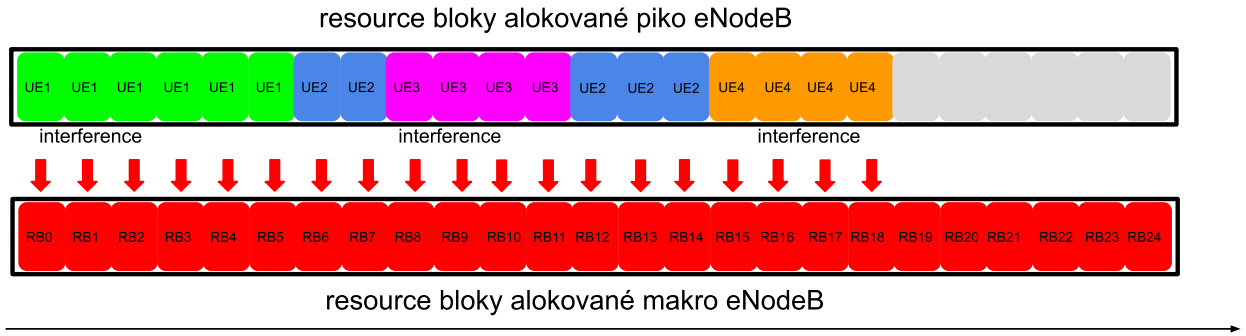
3.2.5.3 *Výstup a zhodnocení výsledků laboratorního cvičení*

Pro vykreslení grafů z výstupních statistik může čtenář použít libovolný nástroj, zde byly grafy statistik opět vykresleny pomocí programovacího jazyka Python s knihovnami matplotlib, numpy, scipy a statsmodels. Čtenáři jsou popsány jednotlivé typy alokací, které jsou k dispozici v SimuLTE frameworku. Po nich následuje ukázka a popis výsledných grafů laboratorního cvičení.



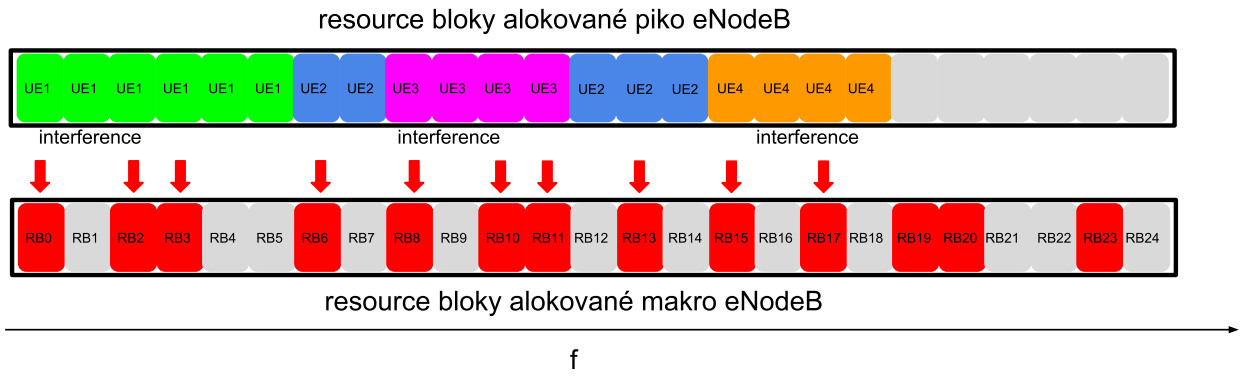
Obrázek 3.8: *LTE rámec, podrámec a slot – 25 RB*

Na obrázku 3.8 je ukázka LTE rámce s 25 resource bloky. Princip alokace je stejný pro libovolný počet resource bloků. V případě režimu plné alokace (*FULL_ALLOC)* dojde k alokování všech resource bloků, které jsou k dispozici v LTE rámci. To znamená, že je stoprocentní pravděpodobnost, že UE v piko buňkách budou na všech resource blocích, které jim jejich obsluhující eNodeB přidělí interferovat se signálem z makro buňky. Se zvyšujícím se vysílacím výkonem makro eNodeB se tedy budou zvyšovat vlivy interference. Obrázek 3.9 ilustruje daný problém.



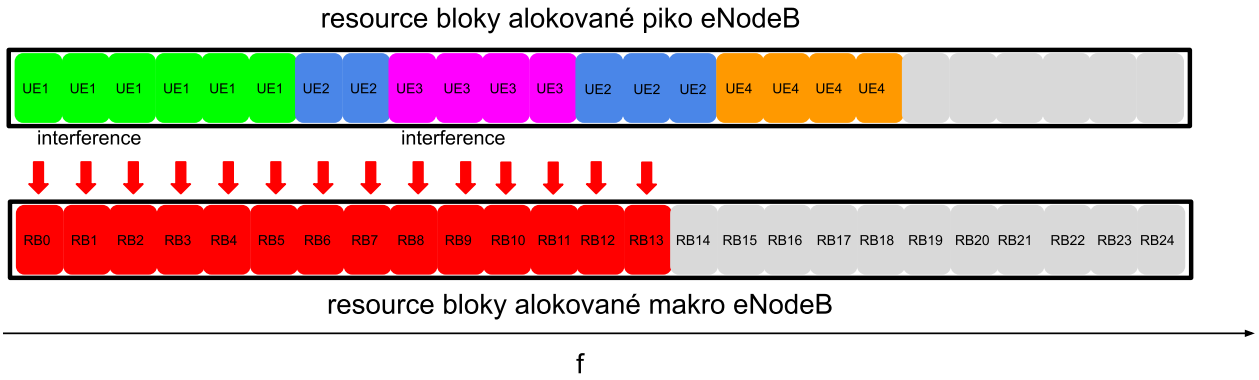
Obrázek 3.9: *Interference při režimu plné alokace*

Obrázek 3.10 je ukázka interference při režimu náhodné alokace (*RANDOM_ALLOC),* kdy dojde k alokování pouze X resource bloků, kde každý resource blok je vybírán náhodně. Počet resource bloků který má být alokován je vypočten pomocí šířky pásma a parametru *bandUtilization,* který je zadáván jako desetinné číslo od 0 do 1. Na rozdíl od plné alokace, zde už není stoprocentní pravděpodobnost, že UE v piko buňkách budou na všech resource blocích interferovat s resource bloky z makro buňky. Bloky jsou vybírány náhodně, tudíž se může stát, že budou alokovány ty bloky, které nebudou odpovídat blokům přiděleným v piko buňce a tím pádem nedojde k interferenci. V krajním případě se může stát, že bude interference minimální – to ovšem vysoce záleží na nastavení parametru *bandUtilization* a náhodnosti přiřazených resource bloků.

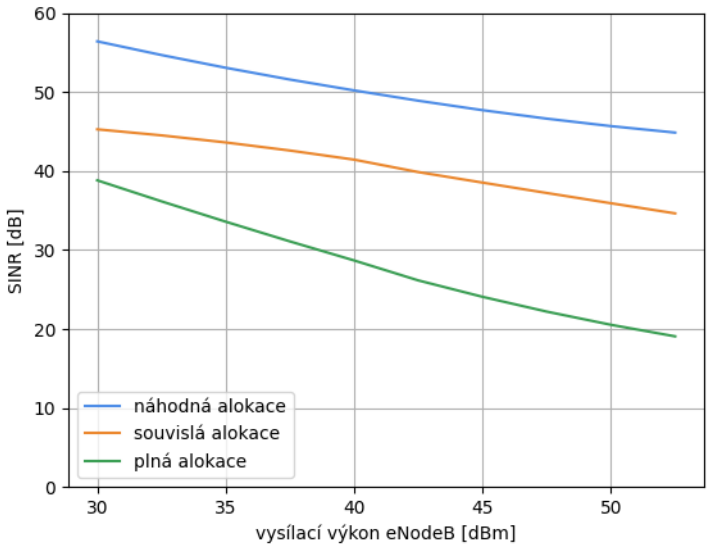


Obrázek 3.10: *Interference při režimu náhodné alokace a 50 % alokovaných resource bloků*

Obrázek 3.11 ukazuje interferenci při režimu souvislé alokace (*CONTIGUOUS_ALLOC),* kdy dojde k alokování celého souvislého bloku X resource bloků, které začínají od specifikovaného resource bloku. Pakliže není začáteční resouce blok definován, začínají od nultého resource bloku. Celkový počet resource bloků, který má být alokován je vypočten stejným principem, jako tomu bylo v případě náhodné alokace. Stejně jako u náhodné alokace zde není stoprocentní pravděpodobnost, že UE v piko buňkách budou na všech resource blocích interferovat s resource bloky z makro buňky. Může se stát, že některé resource bloky přidělené piko UE mohou být mimo jim odpovídajícím resource blokům v souvislém alokovaném bloku, který byl přidělen makro buňkou, a tudíž bude docházet k menší interferenci. Pokud samozřejmě dojde k nastavení parametru *bandUtilization* na hodnotu 0, interference bude dosahovat minimální hodnoty.

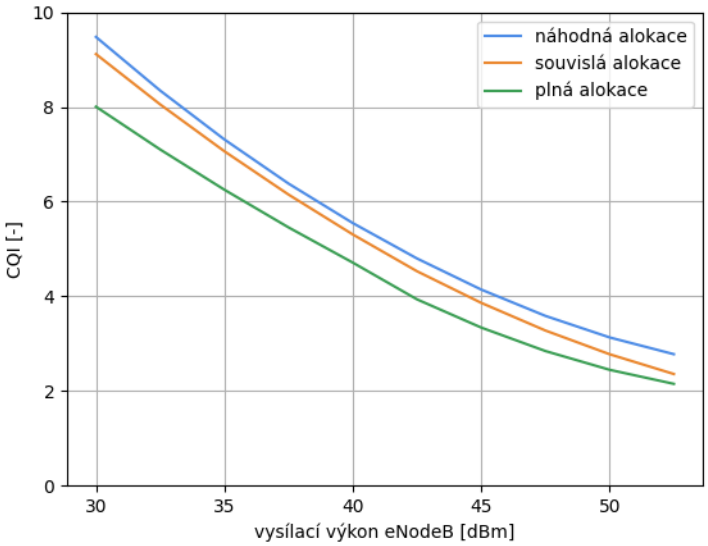


Obrázek 3.11: *Interference při režimu souvislé alokace a 50 % alokovaných resource bloků*



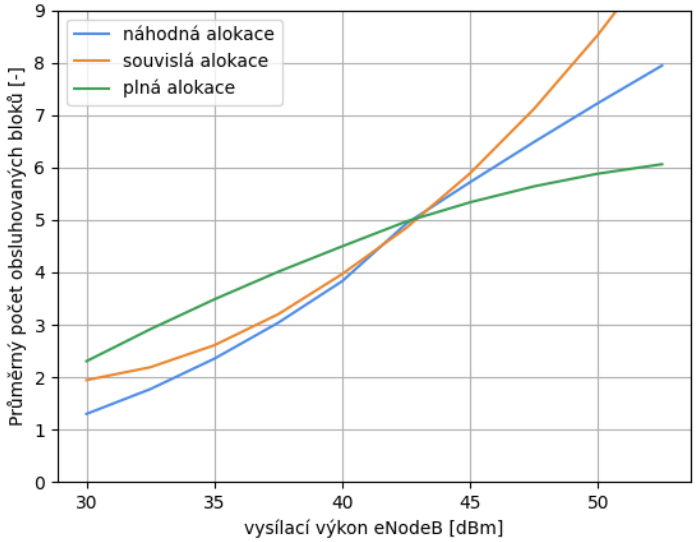
Obrázek 3.12: *Závislost SINR na vysílacím výkonu makro eNodeB*

Poměr signálu k interferenci a šumu (signal to interference plus noise ratio – SINR) slouží ke kvantifikaci vztahu mezi radiofrekvenčními podmínkami a propustností. Jednotkou SINR jsou decibely. Jedná se o sílu požadovaného signálu v poměru s nežádoucím šumem a interferencí. SINR není ovšem definován ve specifikacích 3GPP. Bývá definován dodavateli UE. Optimalizace SINR tedy vede k větší kapacitě základnové stanice, kvalitnější zkušenosti uživatele, umožňuje modulaci QAM vyššího řádu, což má za následek vyšší přenosové rychlosti a méně přerušených hovorů. Do hodnoty SINR se promítají všechny typy rušení. Z obrázku 3.12 je patrné, že s rostoucím vysílacím výkonem makro buňky dochází u všech typů alokací k poklesu SINR. Čím větším vysílacím výkonem makro buňka vysílá, tím k větší interferenci na piko buňkách dochází, což následně snižuje hodnotu SINR. V případě plné alokace dochází k největší interferenci, a to z toho důvodu, že všechny resource bloky které jsou na piko buňkách přiděleny jednotlivým UE interferují s resource bloky na makro buňce – hodnota interference zde bude ze všech typů alokací nejvyšší. U souvislého typu alokace je hodnota SINR menší než u náhodného typu alokace. Je to z toho důvodu, že u souvislého typu alokace se alokuje celý souvislý blok resource bloků, je zde tedy větší pravděpodobnost (v porovnání s náhodnou alokací), že dojde k interferenci některého resource bloku mezi piko UE a makro buňkou. U náhodného typu alokace je hodnota SINR největší, bloky jsou na makro buňce alokovány náhodně – je zde nejmenší šance alokování stejného resource bloku, který je alokován piko buňkou – jelikož jde o náhodnou alokaci.



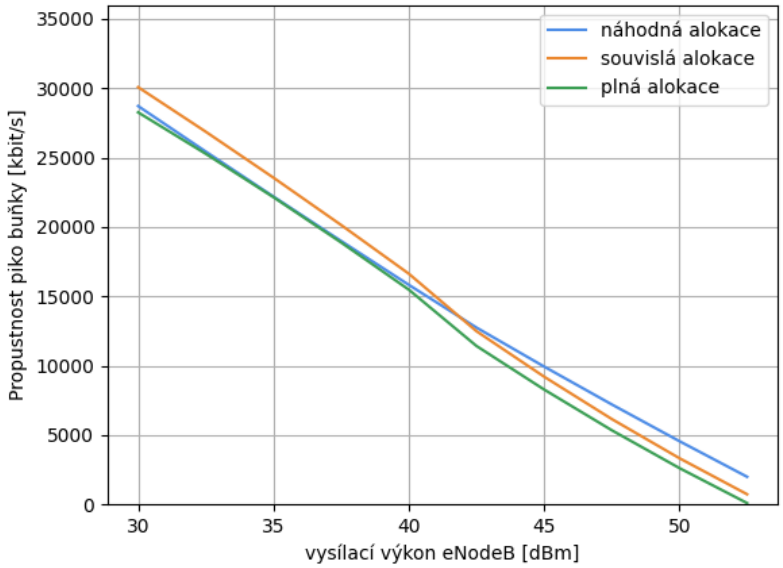
Obrázek 3.13: *Závislost CQI na vysílacím výkonu makro eNodeB*

Obrázek 3.13 je graf identifikátoru popisující kvalitu kanálu (channel quality identifier) v závislosti na vysílacím výkonu makro buňky. Tabulka hodnot CQI je k dispozici v tabulce 3.2. [29] Jak již název napovídá, jedná se o identifikátor nesoucí informaci o tom, jak dobrá/špatná je kvalita komunikačního kanálu. Z grafu je patrné, že s rostoucím vysílacím výkonem makro buňky dochází ke zhoršení rádiových podmínek u všech typu alokací. Nejmenší hodnota CQI nastává u plné alokace, kdy dochází k interferenci na všech resource blocích na kterých piko UE vysílají, což má za následek nejhorší rádiové podmínky komunikačního kanálu. Souvislá alokace vykazuje menší hodnoty CQI než náhodná alokace, kvůli alokaci celého souvislého bloku, čímž dochází k větší pravděpodobnosti k interferenci. Rozdíl v hodnotách CQI mezi souvislou a náhodnou alokaci je malý, avšak od hodnoty vysílacího výkonu 45 dBm se začíná hodnota identifikátoru popisujícího kvalitu kanálu pro souvislou alokaci přibližovat hodnotám pro plnou alokaci. Čím větší vysílací výkon makro buňky, tím k větší interferenci dochází, což má za následek celkové snížení číselné hodnoty identifikátoru.



Obrázek 3.14: *Závislost průměrného počtu obsluhovaných bloků na vysílacím výkonu makro eNodeB*

Třetí graf (obrázek 3.14) říká, kolik průměrně resource bloků je využíváno v rámci piko buňky v síti. V SimuLTE je tento parametr pojmenován jako průměrný počet obsluhovaných bloků. Se zvyšujícím se využitím resource bloků buněk roste pravděpodobnost, že zdroje nemusí být jednotlivým UE přiděleny v čas a spolehlivě. Z obrázku je patrné, že se zvyšujícím se vysílacím výkonem makro buňky dochází ke zvětšení využití obsluhovaných bloků buněk, toto tvrzení platí pro všechny typy alokací. To je způsobeno tím, že při zvětšení vysílacího výkonu makro buňky dochází k větším hodnotám interference a pro požadovaný stejný přenos dat je potřeba použít (alokovat) více bloků. Tento jev je nejzřetelnější u souvislé alokace, zatímco nejméně zřetelný je u plné alokace. To je zapříčiněno tím, že u plné alokace dochází k nejvyšším hodnotám rušení v piko buňkách.



Obrázek 3.15: *Závislost propustnosti piko buňky na vysílacím výkonu makro eNodeB*

Obrázek 3.15 je graf závislosti propustnosti piko buňky na vysílacím výkonu makro eNodeB. Propustnost buňky je v SimuLTE frameworku definována jako parametr *CellThroughput.* Za předpokladu, že všechny ostatní parametry simulace jsou stejné, silnější síla vysílacího výkonu na makro buňce znamená zvětšení interference a zmenšení propustnosti na piko buňkách. Propustnosti piko buněk při jednotlivých typech alokací jsou velice podobné. Nejmenší propustnost při nejvyšších hodnotách vysílacího výkonu je dosažena při plné alokaci, zatímco největší propustnost je dosažena při náhodné alokaci. To je zapříčiněno stejnými důvody jako v předchozích třech případech. Jelikož při plné alokaci dochází k největším hodnotám interference, je propustnost při tomto typu alokace nejmenší, naopak je tomu u náhodné alokace. Při nejvyšší nastavené vysílací výkonové úrovni na makro eNodeB je možné si všimnout, že propustnost piko buňky je velice blízko nulové hodnotě z důsledku velké úrovně interference.

3.3 Ns-3 – Handover v síti LTE

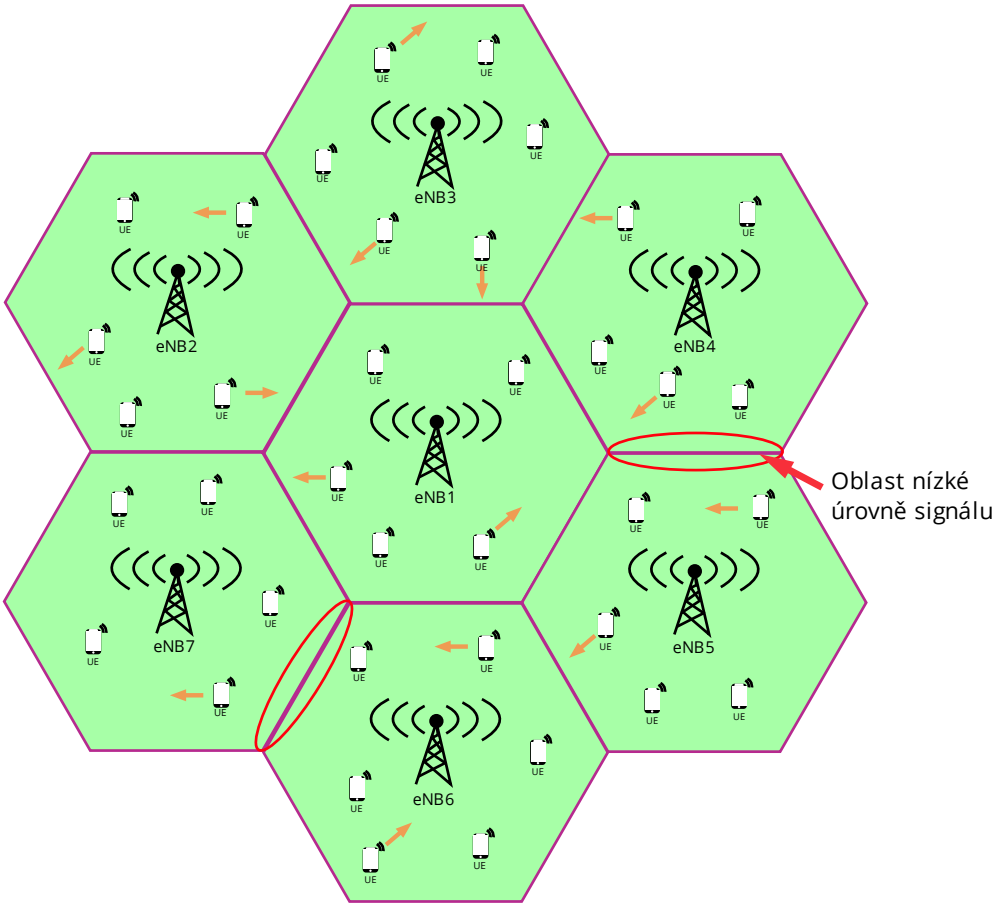
3.3.1 Cíl laboratorního cvičení

Cílem laboratorního cvičení je demonstrace funkcionalit simulačního nástroje ns-3 a modulu LTE čtenáři. Vytvoření sítě obsahující prvky architektury LTE v simulačním nástroji ns-3, demonstrace, pochopení handoveru a popsání změny určitých parametrů v závislosti na rychlosti UE v LTE síti.

3.3.2 Zadání

Před zpracováním laboratorního cvičení se seznamte s problematikou handoveru a synchronizace v LTE sítích. Vytvořte novou simulaci v simulačním nástroji ns-3 a modulu LTE. Vytvořte nový .cc (C++) soubor, a v něm sestavte celou architekturu sítě LTE podle obrázku 3.16 v části architektura sítě. Ve stejném souboru definujte požadovanou funkcionalitu simulace, dle parametrů uvedených v tabulce 3.4. Vzdálenosti mezi eNodeB nastavte tak, aby mezi některými vznikla oblast nízké úrovně signálu – pro využití možnosti sledování selhání rádiového spojení v ns-3 simulátoru. Proveďte simulaci LTE sítě s tím, že v každém kroku simulace zvyšujte rychlost UE o 25 km/h se začátkem v 0 km/h a koncem v 200 km/h. Pohyb a pozice UE budou náhodné. Z výstupních statistik simulace si vyberte čtyři vhodné, rozdílné statistiky a jejich výsledky v závislosti na rychlosti UE vykreslete do grafů. Výsledné grafy popište.

3.3.3 Architektura sítě



Obrázek 3.16: *Architektura sítě LTE pro třetí laboratorní cvičení*

3.3.4 Parametry simulace

Tabulka 3.4: *Parametry třetí simulace*

PARAMETR	HODNOTA
Počet resource bloků	25
Šířka pásma	5 MHz
EPC	Povolen režim EPC
Model mobility	RandomWalk2dMobilityModel (ns3)
Vysílací výkon eNB	40 dBm
Výška eNB	25 m
Vysílací výkon UE	23 dBm
Šum	5 dB
Rychlost UE	0, 25, 50, 75, 100, 125, 150, 175, 200 km/h
Doba simulace	30 sekund
Počet UE na eNB	10
Počet eNB	7
Ideální RRC protokol	ano
Handover algoritmus	A3RsrpHandoverAlgorithm (ns3)
Ztrátový model	ItuR1411NlosOverRooftopPropagationLossModel
Výška budov	20 m
Šířka ulic	20 m
Rozsah budov	2000 m

3.3.5 Postup řešení

3.3.5.1 *Implementace funkcionality sítě*

V adresáři, kde je nainstalován ns-3 framework ve složce *../NS-3.35/scratch* vytvořte nový soubor s názvem *simulationHandover.cc.* Tento soubor slouží pro definování jak celé architektury sítě, tak i její funkcionality. Otevřete si libovolný editor C++ zdrojových souborů (například Microsoft Visual Studio Code) a zkopírujte si základní strukturu zdrojového souboru včetně include příkazů z libovolného ns-3 LTE příkladu, který je k nalezení ve složce *../NS-3.35/src/lte/examples.* Jelikož veškeré příklady a návody pro modul LTE jsou psány do funkce *main,* je vhodné zdrojový kód psát stejným způsobem. Veškeré třídy používané v tomto návodu se nacházejí ve jmenném prostoru *ns3.* První částí je nastavení obecných parametrů týkajících se LTE sítě. Jedná se o model šíření, počet resource bloků (šířka pásma), vysílací výkon, počet základnových stanic, počet uživatelů apod. Z tohoto důvodu definujte globální statické proměnné typu *GlobalValue*. Je vhodné před všechny globální proměnné přidávat předponu ***g_****název_proměnné* z důvodu lepší rozpoznatelnosti od ostatních proměnných. Proměnné budou definovány pro tyto obecné parametry: počet UE – integer, počet eNodeB – integer, počet datových toků pro UE – integer, vysílací výkon eNodeB a UE – double, zapnutí chybového modelu kontrolního a datového kanálu – boolean, použití ideálního RRC protokolu – boolean, šum na fyzické vrstvě UE a eNodeB – double, počet resource bloků – integer, výška budov – double, vzdálenost budov – double, šířka ulic – double. Ve funkci *main* uložte globální proměnné do jejich odpovídajících proměnných. Vytvořte proměnné pro rychlost uživatelů a čas simulace. Lze se inspirovat zdrojovým kódem níže:

```
// ukázka vytvoření dvou globálních proměnných pro počet UE a vysílací výkon UE
static ns3::GlobalValue g_numberOfUes ("numberOfUes",
                                       "Počet celkových UEs v simulaci.",
                                        ns3::UintegerValue (70),
                                        ns3::MakeUintegerChecker<uint16_t> ());
```

```
static ns3::GlobalValue g_ueTxPowerDbm ("ueTxPowerDbm",
                                        "Vysílací výkon UEs v simulaci [dBm].",
                                        ns3::DoubleValue (23.0),
                                        ns3::MakeDoubleChecker<double> ());
int main (int argc, char *argv[])
{
DoubleValue doublevalue;
UintegerValue uintegervalue;
// přiřazení globálních proměnných
GlobalValue::GetValueByName ("numberOfUes", uintegervalue);
uint16_t numberOfUes = uintegervalue.Get();
GlobalValue::GetValueByName ("ueTxPowerDbm", doublevalue);
double ueTxPowerDbm  = doublevalue.Get();
```

Pomocí funkce *Config::SetDefault* nastavte jako výchozí parametry simulace tyto: použití ideálního RRC protokolu – hodnota true, šum na fyzické vrstvě UE a eNodeB – tabulka 3.4, zapnutí chybového modelu kontrolního a datového kanálu – hodnota true, vysílací výkon UE a vysílací výkon eNodeB – tabulka 3.4. Při použití ideálního RRC se modeluje přenos RRC zpráv z UE do eNodeB ideálním způsobem, bez chyb a bez spotřebovávání jakýchkoli rádiových zdrojů – což je zejména výhodné při zjištění selhání rádiového spojení.

```
Config::SetDefault ("ns3::LteHelper::UseIdealRrc", BooleanValue (idealRRC));
Config::SetDefault ("ns3::LteSpectrumPhy::CtrlErrorModelEnabled",
BooleanValue(ctrlError));
Config::SetDefault ("ns3::LteSpectrumPhy::DataErrorModelEnabled",
BooleanValue(dataError));
Config::SetDefault ("ns3::LteUePhy::NoiseFigure", DoubleValue (uePhyNoise));
Config::SetDefault ("ns3::LteEnbPhy::NoiseFigure", DoubleValue (eNBPhyNoise));
Config::SetDefault ("ns3::LteEnbPhy::TxPower", DoubleValue (enbTxPowerDbm));
Config::SetDefault ("ns3::LteUePhy::TxPower", DoubleValue (ueTxPowerDbm));
```

Pomocí funkce *AddValue* a *Parse* třídy *CommandLine* přidejte možnost měnit rychlosti UE a dobu simulace pomocí příkazového řádku.

```
double speed = 20; // m/s
double simTime = 30;
CommandLine cmd (__FILE__);
cmd.AddValue ("simTime", "Total duration of the simulation (in seconds)", simTime);
cmd.AddValue ("speed", "Speed of the UE (default = 20 m/s)", speed);
cmd.Parse (argc, argv);
```

Vytvořte objekt typu *LteHelper*, který slouží uživateli pro jednodušší práci při vytváření architektury rádiové přístupové sítě LTE. Dále vytvořte objekt typu *pointToPointEpcHelper* s názvem epcHelper. Objekt toho typu slouží v ns-3 pro konfiguraci EPC s propojením typu bod-bod v páteřní síti. Třída *EpcHelper* je abstraktní základní třída, která poskytuje pouze definici API, implementace je delegována na podřízené třídy, aby umožnila různé modely sítě EPC, jako je v tomto případě model typu bod-bod. Nastavte objektu lteHelper atribut epcHelper na nově vytvořený pointToPointEpcHelper. Definujte typ plánovače na *ns3::PfFfMacScheduler.* Správně nastavte ztrátový model dle tabulky 3.4 – typ prostředí

bude *UrbanEnvironment* a velikost města bude *MediumCity.* Dále správně nastavte šířku pásma pro uplink i downlink u atributu *EnbDeviceAttribute* objektu *lteHelper.*

```
//Vytvoření LTE sítě plus parametry pro handover scénář
Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
Ptr<PointToPointEpcHelper> epcHelper = CreateObject<PointToPointEpcHelper> ();
lteHelper->SetEpcHelper (epcHelper);
lteHelper->SetSchedulerType ("ns3::PfFfMacScheduler");
// nastavení ztrátového modelu
lteHelper->SetPathlossModelType
(TypeId::LookupByName("ns3::ItuR1411NlosOverRooftopPropagationLossModel"));
lteHelper->SetPathlossModelAttribute
("Environment",EnumValue(ns3::EnvironmentType::UrbanEnvironment));
lteHelper->SetPathlossModelAttribute ("CitySize", EnumValue(ns3::CitySize::MediumCity));


lteHelper->SetPathlossModelAttribute ("RooftopLevel", DoubleValue(rooftopLevel)); //20 m
lteHelper->SetPathlossModelAttribute ("BuildingsExtend", DoubleValue(buildingsExtend));
lteHelper->SetPathlossModelAttribute ("StreetsWidth", DoubleValue(streetWidth)); //20 m


lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (enbBandwidth)); //5 MHz
lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (enbBandwidth)); //5 MHz
```

Dle tabulky 3.4 správně nastavte typ handover algoritmu. Jedná se o algoritmus založený na hodnotě RSRP. Nastavte algoritmus takovým způsobem, že v případě když UE zjistí, že hodnota RSRP z jiné buňky je o 3 dB větší než jeho aktuální hodnota RSRP, tak dojde k přepnutí na eNodeB s vyšší hodnotou RSRP za 256 milisekund. Lze se inspirovat zdrojovým kódem níže:

```
// nastavení algoritmu pro handover
lteHelper->SetHandoverAlgorithmType ("ns3::A3RsrpHandoverAlgorithm");
lteHelper->SetHandoverAlgorithmAttribute ("Hysteresis", DoubleValue (3.0));
lteHelper->SetHandoverAlgorithmAttribute ("TimeToTrigger",
                                          TimeValue (MilliSeconds (256)));
```

Jelikož je v simulaci používáno evolved packet core, tak dalším krokem je potřeba vytvořit vzdáleného hosta, internetu a směrování internetu směrem k LTE síti. Pro tyto tři problémy využijte zdrojový kód níže:

```
Ptr<Node> pgw = epcHelper->GetPgwNode ();
// Vytvoření vzdáleného hosta
NodeContainer remoteHostContainer;
remoteHostContainer.Create (1);
Ptr<Node> remoteHost = remoteHostContainer.Get (0);
InternetStackHelper internet;
internet.Install (remoteHostContainer);
// Vytvoření internetu
PointToPointHelper p2ph;
p2ph.SetDeviceAttribute ("DataRate", DataRateValue (DataRate ("100Gb/s")));
p2ph.SetDeviceAttribute ("Mtu", UintegerValue (1500));
p2ph.SetChannelAttribute ("Delay", TimeValue (Seconds (0.010)));
```

```
NetDeviceContainer internetDevices = p2ph.Install (pgw, remoteHost);
Ipv4AddressHelper ipv4h;
ipv4h.SetBase ("1.0.0.0", "255.0.0.0");
Ipv4InterfaceContainer internetIpIfaces = ipv4h.Assign (internetDevices);
Ipv4Address remoteHostAddr = internetIpIfaces.GetAddress (1);
// Směrování internetového hostitele (směrem k síti LTE)
Ipv4StaticRoutingHelper ipv4RoutingHelper;
Ptr<Ipv4StaticRouting> remoteHostStaticRouting = ipv4RoutingHelper.GetStaticRouting
(remoteHost->GetObject<Ipv4> ());
remoteHostStaticRouting->AddNetworkRouteTo (Ipv4Address ("7.0.0.0"), Ipv4Mask
("255.0.0.0"), 1);
```

Jednotlivé eNodeB lze shromáždit pomocí třídy *NodeContainer* a její funkce *Add* – parametrem funkce je vektor typu *(x_souřadnice, y_souřadnice, výška)* - a příslušné hodnoty v metrech. Dále pak pro potřeby přidělení pozic z deterministického seznamu určeného uživatelem (*NodeContainer*) slouží třída *ListPositionAllocator.* Jednotlivým eNodeB přiřaďte pozice následovně: eNB1 (x = 800m, y = 850m, h = 25 m); eNB2 (x = 1200m, y = 300m, h = 25 m); eNB3 (x = 1400m, y = 850m, h = 25 m); eNB4 (x = 1200m, y = 1450m, h = 25 m); eNB5 (x = 400m, y = 1450m, h = 25 m); eNB6 (x = 200m, y = 850m, h = 25 m); eNB7 (x = 400m, y = 300m, h = 25 m). Pozice UE budou podle specifikace zadání náhodné.

```
//Vytvoření eNB a UE
NodeContainer ueNodes;
NodeContainer enbNodes;
enbNodes.Create (numberOfEnbs);
ueNodes.Create (numberOfUes);
// Nainstalování pozic eNB
Ptr<ListPositionAllocator> enbPositionAlloc = CreateObject<ListPositionAllocator> ();
Vector eNB1Position (600, 650, 25);
enbPositionAlloc->Add (eNB1Position);
  … … …
```

Dalším nezbytným parametrem pro nastavení mobility, je vymezení maximální oblasti pohybu jednotlivých UE a eNodeB. ENodeB se nepohybují, jejich pozice je konstantní a pro nastavení konstantní pozice existuje v ns-3 model mobility s názvem *ConstantPositionMobilityModel*. Nastavení modelu mobility se provádí pomocí funkce *SetMobilityModel,* poté zavoláním funkce *SetPositionAllocator* a poskytnutím daného alokátoru pozic a následným zavoláním funkce *Install* třídy *MobilityHelper.* Vymezení pohybu nastavte na celou oblast simulace tj. 0m < x < 1600m, 0m < y < 1600m. Pro nastavení mobility UE použijte stejné funkce stejného objektu jako v případě eNodeB. Při nastavování modelu mobility u UE je nutné nejdříve nastavit jejich výchozí pozici v simulaci, v opačném případě by objekt typu *MobilityHelper* neměl inicializovány jejich pozice a došlo by k chybě při běhu simulace. Výchozí pozici UE lze nastavit pomocí alokátoru pozic typu *GridPositionAllocator* a příslušných parametrů, které jsou k dispozici ve zdrojovém kódu níže:

```
// Nainstalování modelu mobility na eNB
MobilityHelper enbMobility;
enbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
enbMobility.SetPositionAllocator (enbPositionAlloc);
enbMobility.Install (enbNodes);
```

```
// nainstalování modelu mobility na UE
MobilityHelper ueMobility;
//Inicializace UEs pro pozice uvnitř souřadnic simulace
ueMobility.SetPositionAllocator ("ns3::GridPositionAllocator",
                                 "MinX", DoubleValue (200.0),
                                 "MinY", DoubleValue (0.0),
                                 "DeltaX", DoubleValue (5.0),
                                 "DeltaY", DoubleValue (20.0),
                                 "GridWidth", UintegerValue (20),
                                 "LayoutType", StringValue ("RowFirst"));
// Nastavení modelu mobility na UE
ueMobility.SetMobilityModel ("ns3::RandomWalk2dMobilityModel",
                             "Mode", StringValue ("Time"),
                             "Time", StringValue ("2s"),
                             "Speed", StringValue
("ns3::ConstantRandomVariable[Constant=" + std::to_string(speed) +"]"),
                             "Bounds", StringValue ("0|1600|0|1600"));
ueMobility.Install (ueNodes);
```

Dalším krokem je definice skutečných pozic jednotlivých UE. Pozice UE mají být v simulaci náhodné, k tomuto problému je v ns-3 k dispozici třída *UniformRandomVariable* a její patřičné atributy *Min* a *Max*, pro nastavení minimální a maximální meze, ve které má být náhodná hodnota vygenerována a metody *SetAttribute* a *GetValue.* Pozici UE nastavte buď zcela náhodně nebo náhodně v rámci jim příslušících eNodeB. Ukázka nastavení náhodných pozic pro prvních deset UE v rámci prvního eNodeB je na zdrojovém kódu níže:

```
// Instalace pozice na UE
Ptr<UniformRandomVariable> uePositionX = CreateObject<UniformRandomVariable> ();
uePositionX->SetAttribute ("Min", DoubleValue (700));
uePositionX->SetAttribute ("Max", DoubleValue (900));
Ptr<UniformRandomVariable> uePositionY = CreateObject<UniformRandomVariable> ();
uePositionY->SetAttribute ("Min", DoubleValue (750));
uePositionY->SetAttribute ("Max", DoubleValue (950));
uint16_t currentEnb = 1;
uint16_t numberOfUePerEnb = (numberOfUes/numberOfEnbs) * currentEnb;
for( uint16_t i = 0; i < numberOfUePerEnb; i++){
    ueNodes.Get(i)->GetObject<MobilityModel> ()->SetPosition (Vector (uePositionX->
GetValue(), uePositionY->GetValue(), 1.5));
  }
```

Následující kroky pro vytvoření simulace jsou: instalace LTE zařízení do eNB a UE, instalace IP stacku na UE a připojení všech UE k eNodeB. Instalace LTE zařízení se provádí pomocí *NetDeviceContainer* a metod *InstallDevice* lteHelperu. Pro instalaci IP stacku je potřeba přiřadit IP adresy jednotlivým UE. Pro připojení UE k eNodeB se používá metoda *Attach* třídy *LteHelper.* Parametry metody jsou požadovaný UE a jemu příslušící eNodeB. Proveďte tyto výše popsané kroky.

```
// Instalace LTE zařízení do eNB a UE
NetDeviceContainer enbLteDevs = lteHelper->InstallEnbDevice (enbNodes);
NetDeviceContainer ueLteDevs = lteHelper->InstallUeDevice (ueNodes);
```

```
// Instalace IP stacku na UE
internet.Install (ueNodes);
Ipv4InterfaceContainer ueIpIfaces;
ueIpIfaces = epcHelper->AssignUeIpv4Address (NetDeviceContainer (ueLteDevs));
numberOfUePerEnb = numberOfUes/numberOfEnbs;
// Připojení všech 70 UE k 7 eNodeBs
for(uint16_t i = 0; i < numberOfEnbs; i++ ){
  for (uint16_t j = 0; j < numberOfUePerEnb; j++){
    lteHelper->Attach (ueLteDevs.Get (i*numberOfUePerEnb+j), enbLteDevs.Get (i));
  }  }
```

Správně nainstalujte a zapněte aplikace na UE a vzdáleném hostovi. Před instalací aplikace je vhodné definovat náhodné začátky časů vysílání bitů. K provedení lze použít třídu *UniformRandomVariable*. Nastavte pro každý UE jeho výchozí bránu pomocí funkce *SetDefaultRoute* třídy *Ipv4StaticRouting*. Počet aplikací, které budou vysílány přes EPC nastavte na hodnotu jedna. Hodnoty downlink portů budou začínat od 10000 a hodnoty uplink portů budou začínat od 20000. Pomocí třídy *ApplicationContainer* nainstalujte požadovaný počet aplikací pro downlink a uplink přenos na jednotlivých UE. Pomocí třídy *EpsBearer* nastavte parametry QoS na úrovni nosiče na *EpsBearer::GBR_CONV_VOICE*. Pomocí funkce *Start* třídy *ApplicationContainer* zapněte aplikace. Ukázková implementace nainstalování a zapnutí aplikací na UE a vzdáleném hostovi je k dispozici v elektronické příloze F. Následujícími částmi simulace jsou přidání X2 rozhraní pro potřeby handoveru, zapnutí trasování zdrojů (nahrávání statistik) a připojení trasovacích zdrojů pro sledované parametry, které nejsou obsaženy ve výchozím trasování zdrojů.

Podle zadání protokolu by si měl čtenář vybrat čtyři vhodné, rozdílné statistiky (trasovací zdroje), jejich průběh vykreslit a popsat. Může se také inspirovat tímto návodem, ve kterém jsou vybrány jako 4 statistiky pro sledování tyto: *sinr* (součástí výstupu metody *EnablePhyTraces*), *averageThroughput* (lze vypočítat pomocí výstupu metody *EnableRlcTraces*), *phySyncDetection, radioLinkFailure* - tento trasovací zdroj je zvláště důležitý pro simulaci, jelikož se projevuje v oblastech nízké úrovně signálu. Důležité je také zmínit fakt, že statistiky by se neměly začít nahrávat od úplného začátku simulace, ale až po tzv. zahřívací periodě. Dobu zahřívací periody nastavte alespoň na hodnotu 300 milisekund. V této době totiž dochází k inicializaci stavu sítě, připojení jednotlivých UE k eNodeB, navázání spojení, selekce buněk a statistiky v tomto časovém intervalu mohou být zkreslené.

```
// Přidání X2 interface
lteHelper->AddX2Interface (enbNodes);
// Zapnutí nahrávání statistik
lteHelper->EnablePhyTraces ();
lteHelper->EnableRlcTraces ();
Ptr<RadioBearerStatsCalculator> rlcStats = lteHelper->GetRlcStats ();
rlcStats->SetAttribute ("StartTime", TimeValue (Seconds (0.3)));  //zahřívací perioda
rlcStats->SetAttribute ("EpochDuration", TimeValue (Seconds (simTime)));
```

Metody *lteHelperu* s prefixem *Enable* a postfixem *Traces* vytváření v kořenovém adresáři simulace nové *.txt* soubory s výsledky simulace. Pro trasovací zdroje pro detekci mimo synchronizaci a selhání rádiového spojení je nutné vytvořit callback metody. Pro připojení těchto trasovacích zdrojů do konfiguračního subsytému použijte tyto konfigurační cesty: *NodeList/*/DeviceList/*/LteUeRrc/RadioL*

inkFailure, *NodeList/*/DeviceList/*/LteUeRrc/PhySyncDetection.* Ukázka implementace callback metody pro libovolný trasovací zdroj a přidání trasovacího zdroje do konfiguračního subsystému je k dispozici v sekci 2.3.2. Callback metody se zavolají v případě, kdy dojde k selhání rádiového spojení, respektive k detekci mimo synchronizaci, definujte si proto vhodné globální proměnné, které budou zachycovat počet takto vzniklých jevů v simulaci – budou se zvyšovat v jím odpovídajícím callback metodách. U trasovacího zdroje pro detekci mimo synchronizaci je důležité si dát pozor na stav. Jelikož tento trasovací zdroj detekuje jak události v synchronizaci, tak mimo synchronizaci. Vhodným porovnáním stavu zachyťte správné události. Trasovací zdroje, které se ve výchozím stavu nezapisují do výstupních souborů, uložte do nového vámi vytvořeného výstupního souboru.

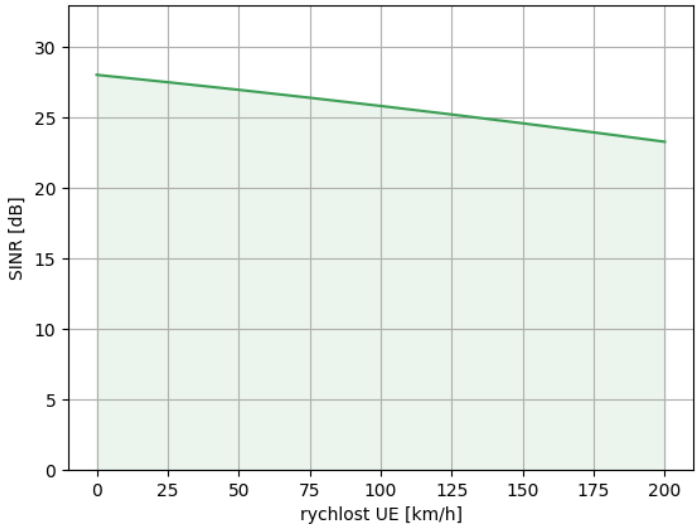
Posledním volitelným krokem je konfigurace *NetAnim*. NetAnim slouží pro grafickou vizualizaci celé simulace. Je dobrým nástrojem pro zjištění, zda uživatel nastavil pozice v simulaci správně a pro obecný přehled celé architektury simulace (pomoc uživateli, aby si nemusel architekturu sítě představovat pouze ze zdrojového kódu). Pro jeho využití ve zdrojovém kódu je nutné přidat hlavičkový soubor *ns3/netanim-module.h*. Ukázka konfigurace NetAnim je k nalezení v sekci 2.3.1.

3.3.5.2 ***Výstup a zhodnocení výsledků laboratorního cvičení***

K zapnutí simulace s rychlostí UE 40 metrů za sekundu (144 km/h) lze použít tento příkaz:

```
./waf --run "simulationHandover --speed=40"
```

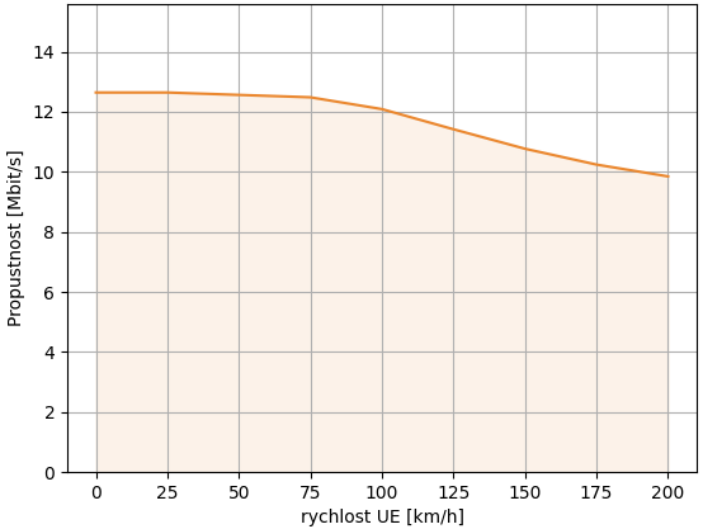
Po zapnutí simulace dojde k vytvoření textových souborů v adresáři, ve kterém je uložen ns-3 simulátor. Pomocí těchto souborů lze výsledky simulace vyexportovat a pomocí libovolného nástroje pro vykreslování grafů vykreslit. Pro vykreslení grafů byl znovu použit programovací jazyk Python s knihovnami matplotlib, numpy, scipy a statsmodels. Čtenář může pro vykreslení grafů použít libovolný vykreslovací nástroj dle svého uvážení. Cílem protokolu je výsledné grafy cvičícímu popsat a ověřit tak, že student chápe souvislosti mezi jednotlivými výstupními statistikami. Následují jednotlivé grafy s teoretickým popisem a vysvětlením.



Obrázek 3.17: *Graf závislosti SINR na rychlosti UE*

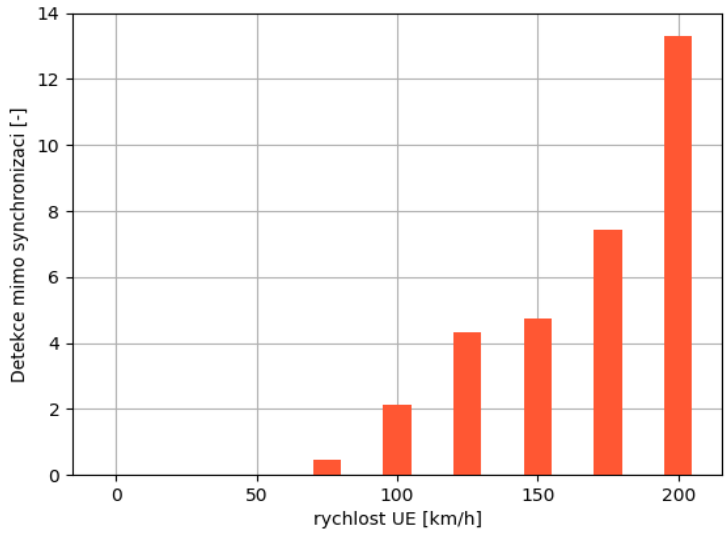
Prvním grafem je graf závislosti SINR (poměr signálu k interferenci a šumu) na rychlosti UE. UE používají SINR k výpočtu CQI, který poté hlásí svému obsluhujícímu eNodeB do sítě. Z obrázku 3.17 je patrné, že s rostoucí rychlostí UE dochází k poklesu SINR. Čím rychleji se jednotliví uživatelé pohybují, tím častěji se UE dostává z oblasti s nízkou úrovní signálu do oblasti s vyšší úrovní signálu (v simulaci existují místa s nízkou úrovní signálu), po různých cestách a působí na něj nežádoucí vlivy. Čím rychleji

se pohybuje, tím častěji na něj tyto jevy působí, což vyústí v celkové zhoršení hodnoty SINR. Jelikož všechny buňky v této simulaci pracují na stejné frekvenci, tak čím více se UE vzdaluje od středu své buňky směrem k okraji, tím více na něj působí interference ze sousedící buňky, což má za následek také pokles hodnoty SINR. Pokles této hodnoty není natolik velký, a i při rychlosti 200 km/h mají UE více než dostačující hodnotu SINR.



Obrázek 3.18: Graf závislosti propustnosti na rychlosti UE

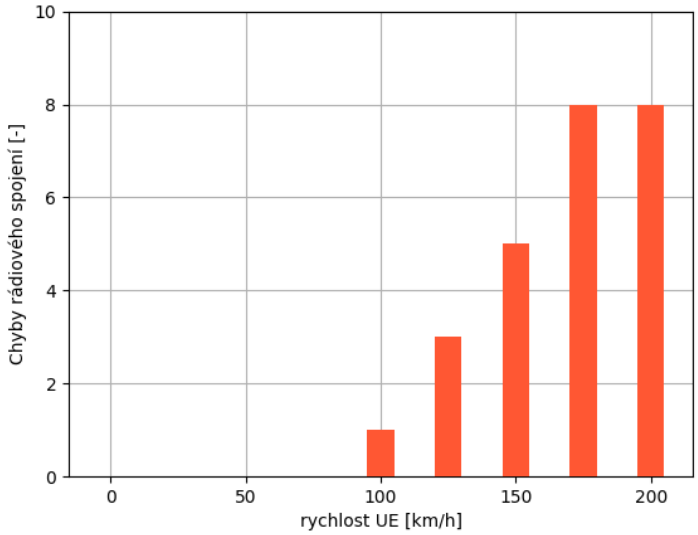
Druhým grafem je graf závislosti průměrné propustnosti na rychlosti UE. Z grafu jde vidět, že s vyšší rychlostí uživatelů dochází ke zmenšení propustnosti. Vysoká rychlost také zvyšuje počet handoverů a reselekcí (opětovný výběr) buněk. Vyšší rychlost dále znamená větší pravděpodobnost, a i častější průchod oblastmi s nízkou úrovní rádiového signálu v simulaci, což se samozřejmě může projevit i na menší hodnotě propustnosti. Nejmenší propustnost je dosažena při nejvyšší rychlosti, zatímco největší propustnost při nulové rychlosti. Ns-3 simulátor jako výstup udává hodnotu přenesených bajtů za celou dobu simulace, je tedy nutné hodnotu správně vypočítat a převést.



Obrázek 3.19: Graf závislosti počtu detekcí mimo synchronizaci za sekundu na rychlosti UE

Třetím grafem je graf závislosti počtu detekcí mimo synchronizaci na rychlosti UE. Graf ukazuje, že s vyšší rychlostí uživatelů dochází k většímu počtu detekcí mimo synchronizaci. V LTE existují dva synchronizační signály pro downlink, které UE používá k získání id buňky a časování rámce. Jedná se o primární synchronizační signál (PSS) a sekundární synchronizační signál (SSS). S rostoucí rychlostí

uživatelů je těžší uživatele synchronizovat a dochází k většímu počtu detekcí mimo synchronizaci – což znázorňuje popisovaný graf (obrázek 3.19). Detekce mimo synchronizaci je základní interní procedura UE, která je zahájena, když mobilní telefon nepřijímá žádné informace o downlinku. To může být způsobeno příliš velkými rychlostmi uživatelů nebo například jako v simulaci uměle vytvořenými oblastmi s nízkou úrovní signálu. Počet těchto detekcí roste z důvodu častějšího průchodu těmito oblastmi a vyššími rychlostmi UE. S detekcí mimo synchronizaci se v ns-3 pojí tři časovače: n310, t310, n311. Časovač n310 určuje maximální počet detekcí mimo synchronizaci, jeho výchozí hodnota je v ns-3 simulátoru nastavena na 4. Časovač t310 slouží pro detekci selhání rádiového spojení, respektive slouží k tomu, aby se UE dostalo zpět do synchronizace s eNodeB. Poslední časovač udává maximální počet detekcí v synchronizaci, jeho výchozí hodnota je nastavena na 2. Pokud UE detekuje n310 po sobě jdoucích indikací mimo synchronizaci, spustí se časovač t310. Pokud časovač vyprší, spojení selhalo. Pokud UE detekuje n311 po sobě jdoucí indikací v synchronizaci před vypršením časovače t310, pak se časovač zastaví a spojení neselže – je úspěšné.



Obrázek 3.20: *Graf závislosti počtu selhání rádiového spojení na rychlosti UE*

Poslední graf popisuje závislost počtu selhání rádiového spojení na rychlosti UE. Je nutné zmínit, že ns-3 na rozdíl od simulačního frameworku SimuLTE poskytuje monitorování selhání rádiového spojení. Z grafu je patrné, že s rostoucí rychlostí UE dochází k nárůstu počtu selhání rádiového spojení. Důvodem je, že UE prochází častěji oblastmi nízké úrovně signálu což má za následek to, že uživatel nepřijímá žádné informace o downlinku a může dojít k selhání rádiového spojení. V ns-3 simulátoru se může UE synchronizovat (tj. začít číst systémové informace) s eNodeB na nízké úrovni RSRP, která lze nastavit pomocí atributu *QRxLevMin* eNodeB RRC, což umožňuje UE zahájit proceduru náhodného přístupu s eNodeB. V případě použití ideálního RRC protokolu může UE dokončit proceduru navázání spojení RRC bez jakýchkoliv chyb, protože dochází k výměně všech zpráv. Na druhou stranu při vypnutí použití ideálního RRC protokolu nebude UE po selhání rádiového spojení moci navázat spojení s eNodeB.

3.4 Ns-3 – Interference ve dvouvrstvé LTE síti

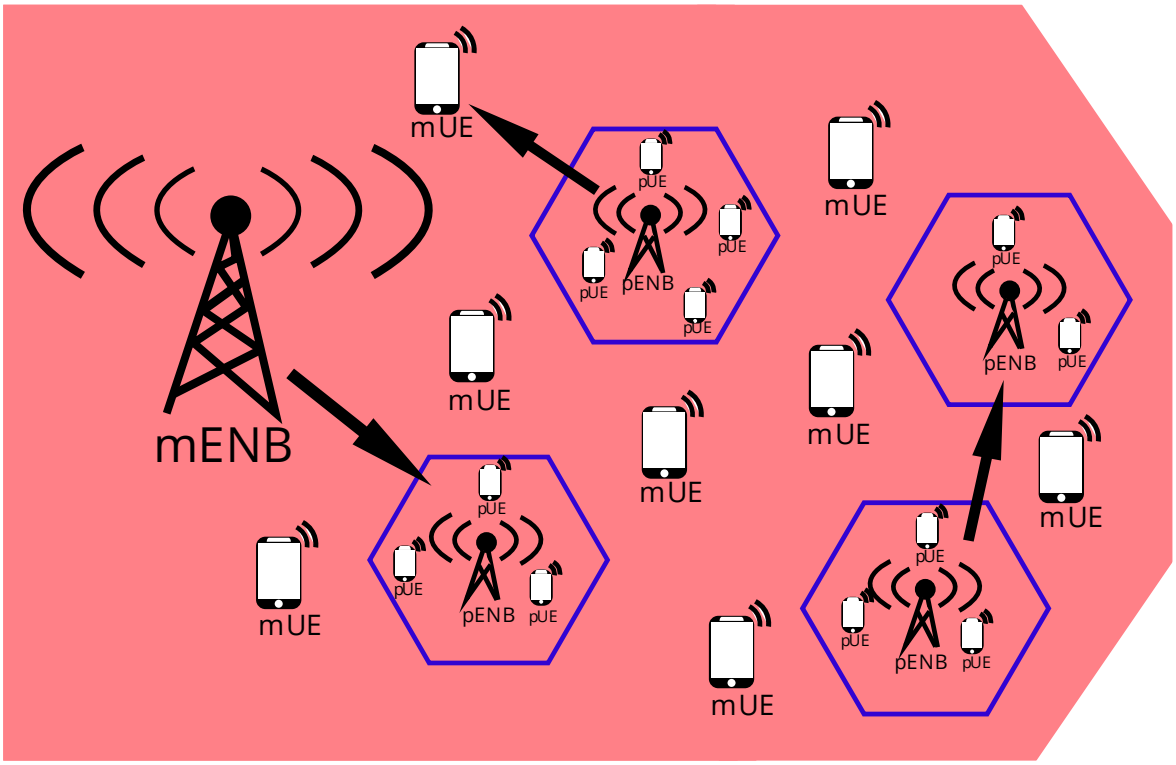
3.4.1 Cíl laboratorního cvičení

Cílem laboratorního cvičení je demonstrace funkcionalit simulačního nástroje ns-3 a modulu LTE čtenáři. Vytvoření sítě obsahující prvky architektury LTE v simulačním nástroji ns-3, demonstrace a pochopení problému interference v síti LTE. Popsání změny určitých parametrů v piko buňkách v závislosti na vysílacím výkonu makro buňky a porovnání různých typů algoritmů pro opětovné použití frekvencí.

3.4.2 Zadání

Před zpracováním laboratorního cvičení se seznamte s problematikou interference a algoritmy pro opětovné použití frekvencí v LTE sítích. Vytvořte novou simulaci v simulačním nástroji ns-3 a modulu LTE. Vytvořte nový *.cc* (C++) soubor a v něm sestavte celou architekturu sítě LTE podle obrázku 3.21 v části architektura sítě. Ve stejném souboru definujte požadovanou funkcionalitu simulace, dle parametrů uvedených v tabulce 3.5. V první části protokolu proveďte simulaci celé LTE sítě s tím, že v každém kroku simulace zvyšujte vysílací výkon makro eNodeB o 2,5 dBm se začátkem v 30 dBm a koncem v 52,5 dBm. Ve druhé části protokolu správně nastavte tři vámi vybrané algoritmy pro opětovné použití frekvencí a pro každý z nich proveďte simulaci celé LTE sítě s tím, že vysílací výkon makro buňky bude nastaven na hodnotu 46 dBm. UE se nebudou pohybovat. Z výstupních statistik první části protokolu si vyberte dvě vhodné, rozdílné statistiky a jejich výsledky v závislosti na vysílacím výkonu makro eNodeB vykreslete do grafů. Z výstupních statistik druhé části protokolu si rovněž vyberte dvě vhodné, rozdílné statistiky a jejich výsledky v závislosti na rozdílných typech algoritmů pro opětovné použití frekvencí vykreslete do grafů. Výsledné grafy popište.

3.4.3 Architektura sítě



Obrázek 3.21: *Architektura sítě pro čtvrté laboratorní cvičení*

Legenda:


- interference

mUE - makro user equipment

pUE - piko user equipment

mENB - makro eNodeB

pENB - piko eNodeB

3.4.4 **Parametry simulace**

Tabulka 3.5: *Parametry čtvrté simulace*

PARAMETR	HODNOTA
Model mobility	**ConstantPositionMobilityModel**
Počet resource bloků	**50**
Šířka pásma	**10 MHz**
Rádius makro buňky	**1000 m**
Rádius piko buňky	**100 m**
Počet pUE v piko buňce	**4**
Počet piko buněk	**4**
Vysílací výkon makro buňky	**30 - 52,5 dBm**
Vysílací výkon piko buňky	**27 dBm**
Vysílací výkon UE	**23 dBm**
Ztrátový model	**ItuR1411NlosOverRooftopPropagationLossModel**
Šířka ulic	**20 m**
Rozsah budov	**2000 m**
Výška budov	**10m**
Čas simulace	**30 sekund**
Algoritmy pro znovupoužití frekvencí	**LteFrNoOpAlgorithm, LteFrHardAlgorithm**
	LteFrStrictAlgorithm, LteFrSoftAlgorithm
	LteFfrSoftAlgorithm

3.4.5 **Postup řešení**

3.4.5.1 ***Implementace funkcionality sítě***

Vytvořte nový soubor s názvem *simulationInterference.cc* ve složce *../NS-3.35/scratch*, která se nachází v adresáři, kde je nainstalován ns-3 framework. V tomto souboru postupně nadefinujte funkcionalitu i celou architekturu LTE sítě. Otevřete si libovolný editor C++ zdrojových souborů a z libovolného ns-3 LTE příkladu si zkopírujte základní strukturu zdrojového souboru včetně include příkazů (příklady jsou k nalezení ve složce *../NS-3.35/src/lte/examples*). Je vhodné se řídit stejným stylem psaní kódu jako v ns-3 příkladech – hlavní funkcionalita a architektura sítě je definována ve funkci *main,* důležité parametry simulace pak jako globální proměnné. Veškeré třídy používané v tomto návodu se nacházejí ve jmenném prostoru *ns3.* První částí je nastavení obecných parametrů týkajících se LTE sítě. Jedná se o model šíření, počet resource bloků (šířka pásma), vysílací výkon, počet základnových stanic, počet uživatelů apod. Na rozdíl od SimuLTE frameworku nemá modul LTE v ns-3 simulátoru žádný uzel typu *externalCell,* tudíž není možné zjednodušit makro buňku stejným způsobem, jako tomu bylo v případě SimuLTE. Makro buňku již nevytvoří framework sám, ale musí si ji definovat uživatel. Makro buňku je nutné realizovat stejným způsobem jako ostatní základnové stanice v simulaci a přiřadit jí patřičné, obsluhované UE. Pro potřeby simulace definujte globální statické proměnné typu *ns3::GlobalValue*. Je vhodné před všechny globální proměnné přidávat předponu ***g_****název_proměnné* z důvodu odlišení od ostatních proměnných. Proměnné budou definovány pro

tyto obecné parametry: počet makro UE – integer, počet piko UE – integer, počet makro buněk – integer, počet piko buněk – integer, vysílací výkon makro buňky – double, vysílací výkon piko buňky – double, vysílací výkon makro a piko UE – double, šum na fyzické vrstvě UE a eNodeB – double, počet resource bloků – integer, výška budov – double, vzdálenost budov – double, šířka ulic – double, zapnutí chybového modelu kontrolního a datového kanálu – boolean, použití ideálního RRC protokolu - boolean. Ve funkci *main* uložte globální proměnné do jejich odpovídajících proměnných. Vytvořte proměnné pro typ algoritmu – *std::string* a čas simulace - double. Lze se inspirovat zdrojovým kódem níže:

```
// ukázka vytvoření dvou globálních proměnných pro počet UE a vysílací výkon UE
static ns3::GlobalValue g_enbBandwidth ("enbBandwidth ",
                                        "Bandwidth [počet RBs] používaný eNB.",
                                         ns3::UintegerValue (50),
                                         ns3::MakeUintegerChecker<uint16_t> ());

static ns3::GlobalValue g_picoTxPowerEnB ("picoTxPowerEnB ",
                                          "Vysílací výkon piko buňky eNB v simulaci.",
                                           ns3::DoubleValue (27.0) //[dBm].,
                                           ns3::MakeDoubleChecker<double> ());
int main (int argc, char *argv[])
{
DoubleValue doublevalue;
UintegerValue uintegervalue;

// přiřazení globálních proměnných
GlobalValue::GetValueByName ("enbBandwidth ", uintegervalue);
uint16_t enbBandwitdth = uintegervalue.Get();

GlobalValue::GetValueByName ("picoTxPowerEnB ", doublevalue);
double picoUETxPowerDbm = doublevalue.Get();
```

Simulátor pro změnu parametrů bez úpravy zdrojového kódu poskytuje subsystém pro ovládání argumentů příkazového řádku. Tento subsystém slouží pro nastavení globálních hodnot (proměnných) a atributů před začátkem běhu simulace. Do toho subsystému patří i pomocná třída *CommandLine.* Pomocí této třídy je možné propojovat proměnné v simulaci s příkazovým řádkem. Pomocí třídy *CommandLine* a její funkce *AddValue* přidejte do simulace možnost měnit proměnné pro dobu simulace, nastavení typu algoritmu pro znovupoužití frekvencí a parsujte patřičné argumenty příkazového řádku.

```
CommandLine cmd (__FILE__);
cmd.AddValue ("simTime", "Total duration of the simulation (in seconds)", simTime);
cmd.AddValue ("frAlgorithm", " Algorithm for frequency reuse (NOOP, HARD, STRICT, SOFT,
SOFT_, SOFTFR)", frAlgorithm);
cmd.Parse (argc, argv);
```

Pro nastavení výchozích parametrů simulace existuje v ns-3 jmenný prostor pro různé funkce implementující systém konfigurací. Tento jmenný prostor nese název *Config.* Existuje zde funkce *SetDefault*, která slouží pro nastavení výchozích parametrů. Funkce má vstupní parametry *std::string*

ns3::parametr a *const AttributeValue & hodnota* – hodnota, která má být přiřazena danému parametru. Pomocí této funkce správně nastavte šum na fyzických vrstvách eNodeB, UE a povolte zapnutí chybového modelu kontrolního a datového kanálu.

```
Config::SetDefault ("ns3::LteSpectrumPhy::CtrlErrorModelEnabled",
BooleanValue(ctrlError));
Config::SetDefault ("ns3::LteSpectrumPhy::DataErrorModelEnabled",
BooleanValue(dataError));
Config::SetDefault ("ns3::LteUePhy::NoiseFigure", DoubleValue (uePhyNoise));
Config::SetDefault ("ns3::LteEnbPhy::NoiseFigure", DoubleValue (eNBPhyNoise));
```

Pro konfiguraci a vytvoření uzlů LTE sítě slouží třída *LteHelper.* Pro simulaci LTE typicky stačí pouze jedna instance třídy *LteHelper.* Obecnou odpovědností tohoto *helperu* je vytvořit různé objekty LTE a propojit je dohromady, aby vznikl celý LTE systém, dále pak konfigurace dodatečných parametrů sítě jako jsou ztrátový model, prostředí, plánovač a šířka pásma. V této simulaci nevytvářejte žádný objekt typu *EpcHelper*, jelikož v této simulaci nebude EPC použito. Definujte typ plánovače na *ns3::PfFfMacScheduler.* Správně nastavte ztrátový model dle tabulky 3.4 – typ prostředí bude *UrbanEnvironment* a velikost města bude *MediumCity.* Dále správně nastavte šířku pásma pro uplink i downlink atributu *EnbDeviceAttribute* objektu *lteHelper.* Povolte použití ideálního RRC protokolu.

```
// Nastavení výchozích parametrů lteHelperu
Ptr<LteHelper> lteHelper = CreateObject<LteHelper> ();
lteHelper->SetAttribute ("UseIdealRrc", BooleanValue (idealRRC));
// nastavení ztrátového modelu
lteHelper->SetPathlossModelType (TypeId::LookupByName
("ns3::ItuR1411NlosOverRooftopPropagationLossModel"));
lteHelper->SetPathlossModelAttribute ("Environment",
EnumValue(ns3::EnvironmentType::UrbanEnvironment));
lteHelper->SetPathlossModelAttribute ("CitySize", EnumValue(ns3::CitySize::MediumCity));


lteHelper->SetPathlossModelAttribute ("RooftopLevel", DoubleValue(rooftopLevel)); //10 m
lteHelper->SetPathlossModelAttribute ("BuildingsExtend", DoubleValue(buildingsExtend));
//2000m
lteHelper->SetPathlossModelAttribute ("StreetsWidth", DoubleValue(streetWidth)); // 20 m
// plánovač a šířky pásem
lteHelper->SetSchedulerType ("ns3::PfFfMacScheduler");
lteHelper->SetEnbDeviceAttribute ("DlBandwidth", UintegerValue (enbBandwidth)); //10 MHz
lteHelper->SetEnbDeviceAttribute ("UlBandwidth", UintegerValue (enbBandwidth)); //10 MHz
```

Typicky objekty v ns-3 pracují na více než jednom uzlu současně, proto se mohou uzly shromažďovat do tak zvaných *NodeContainers.* Ty obsahují více *Node*, které se používají k odkazování na uzly sítě. Jednotlivé eNodeB lze shromáždit pomocí třídy *NodeContainer* a její funkce *Add* – parametrem funkce je vektor typu *(x_souřadnice, y_souřadnice, výška)* - a příslušné hodnoty v metrech. Dále pak pro potřeby přidělení pozic z deterministického seznamu určeného uživatelem (*NodeContainer*) slouží třída *ListPositionAllocator.* Jednotlivým eNodeB přiřaďte pozice následovně: pENB1 (x = 400m, y = 550m, h = 25 m); pENB2 (x = 450m, y = 150m, h = 25 m); pENB3 (x = 650m, y = 200m, h = 25 m); pENB4 (x = 700m, y = 590m, h = 25 m); makroENB (x = 250m, y = 250m, h = 25 m).

```
//Vytvoření eNB a UE
NodeContainer enbNodesMacro, enbNodesPico;
```

```
NodeContainer ueNodesMacro, ueNodesPico;
enbNodesMacro.Create (numberOfMacroEnbs);
enbNodesPico.Create(numberOfPicoEnbs);
ueNodesMacro.Create (numberOfMacroUes);
ueNodesPico.Create (numberOfPicoUes);

// Nainstalování pozic eNB
Ptr<ListPositionAllocator> enbPositionAlloc = CreateObject<ListPositionAllocator> ();

Vector macroENBPosition (250, 250, 25);
enbPositionAlloc->Add (macroENBPosition);
…  …  …
```

Dalším nezbytným parametrem pro nastavení mobility je vymezení maximální oblasti pohybu jednotlivých UE a eNodeB. ENodeB se nepohybují, jejich pozice je konstantní a pro nastavení konstantní pozice existuje v ns-3 model mobility s názvem *ConstantPositionMobilityModel*. Nastavení modelu mobility se provádí pomocí funkce *SetMobilityModel,* poté zavoláním funkce *SetPositionAllocator* a poskytnutím daného alokátoru pozic a následným zavoláním funkce *Install* třídy *MobilityHelper.* Třída *MobilityHelper* slouží pro přidělení pozic a modelu mobility jednotlivým uzlům sítě. Pro nastavení mobility UE použijte stejné funkce stejného objektu jako v případě eNodeB. Jelikož se v této simulaci UE nepohybují, nastavte jim stejný model mobility jako v případě eNodeB. Inspirace je poskytnuta pomocí zdrojového kódu níže:

```
// Instalace mobility modelu v eNB
MobilityHelper enbMobility;
enbMobility.SetMobilityModel ("ns3::ConstantPositionMobilityModel");
enbMobility.SetPositionAllocator (enbPositionAlloc);
enbMobility.Install (enbNodesMacro);
enbMobility.Install (enbNodesPico);
…  …  …
```

Umístěte UE příslušící daným piko eNodeB do kruhové oblasti s maximálním poloměrem 100 metrů. Středovým bodem budou souřadnice obsluhujícího eNodeB. UE které budou obsluhovány makro eNodeB přidělte náhodnou pozici v rámci celé oblasti simulace tj. 200m < x < 1000m, 0m < y < 600m. Pro generování náhodných souřadnic UE použijte funkci *GetValue* třídy *UniformRandomVariable.* Poté nastavte náhodné pozice jednotlivým UE pomocí metody *SetPosition* třídy *MobilityModel.* Jedná se o základní třídu pro všechny specifické modely mobilit, ta sleduje aktuální polohu a rychlost uzlu v síti a veškeré jednotky udržuje v metrech.

```
// Přidělení pozice na UE
Ptr<UniformRandomVariable> uePositionX = CreateObject<UniformRandomVariable> ();
uePositionX->SetAttribute ("Min", DoubleValue (200));
uePositionX->SetAttribute ("Max", DoubleValue (1000));
Ptr<UniformRandomVariable> uePositionY = CreateObject<UniformRandomVariable> ();
uePositionY->SetAttribute ("Min", DoubleValue (000));
uePositionY->SetAttribute ("Max", DoubleValue (600));

for( uint16_t i = 0; i < numberOfMacroUes; i++){
```

```
    ueNodes.Get(i)->GetObject<MobilityModel> ()->SetPosition (Vector (uePositionX->
GetValue(), uePositionY->GetValue(), 1.5));
  }
 … … …
```

Dalším krokem je vytvoření *NetDeviceContainer* pro makro a piko eNodeB, z důvodu vytvoření zařízení a jejich instalaci v uzlech. Tento kontejner vytvoří pro každý uzel instanci *net* zařízení, přidá MAC adresu a nainstaluje je patřičnému uzlu. Každé zařízení je přidáno do kontejneru pro pozdější použití volajícím. Zdrojový kód níže popisuje vytvoření *NetDeviceContainer* pro makro a piko eNodeB a ukázkovou konfiguraci dvou algoritmů pro znovupoužití frekvencí, konkrétně se jedná o algoritmy *LteFrNoOpAlgorithm* a *LteFrHardAlgorithm*. Ukázkové konfigurace zbylých algoritmů z tabulky 3.5 jsou umístěny v elektronické příloze G. Proveďte konfiguraci vybraných tří algoritmů pro opakované použití frekvencí. Všechny algoritmy, ze kterých si čtenář může vybrat, jsou popsány v sekci 3.4.5.2.

```
// Vytvoření zařízení a jejich instalace v uzlech (eNB a UE)
NetDeviceContainer enbMacroDevs;
NetDeviceContainer enbPicoDevs;

// Konfigurace jednotlivých algoritmů pro opakované využití frekvencí
if (frAlgorithm == "NOOP"){
    lteHelper->SetFfrAlgorithmType ("ns3::LteFrNoOpAlgorithm");
    enbMacroDevs = lteHelper->InstallEnbDevice (enbNodesMacro);
    enbPicoDevs = lteHelper->InstallEnbDevice(enbNodesPico);
}
else if (frAlgorithm == "HARD"){

    lteHelper->SetFfrAlgorithmType ("ns3::LteFrHardAlgorithm");
    // Rozdělení 50 resource bloků mezi všechny eNB -> 18 má makro, piko mají po 8
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandOffset", UintegerValue (0));
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandwidth", UintegerValue (18));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandOffset", UintegerValue (0));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandwidth", UintegerValue (18));
    enbMacroDevs.Add(lteHelper->InstallEnbDevice (enbNodesMacro));

    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandOffset", UintegerValue (18));
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandwidth", UintegerValue (8));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandOffset", UintegerValue (18));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandwidth", UintegerValue (8));
    enbPicoDevs.Add(lteHelper->InstallEnbDevice (enbNodesPico.Get(0)));

    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandOffset", UintegerValue (26));
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandwidth", UintegerValue (8));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandOffset", UintegerValue (26));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandwidth", UintegerValue (8));
    enbPicoDevs.Add(lteHelper->InstallEnbDevice (enbNodesPico.Get(1)));

    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandOffset", UintegerValue (34));
```

```
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandwidth", UintegerValue (8));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandOffset", UintegerValue (34));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandwidth", UintegerValue (8));
    enbPicoDevs.Add(lteHelper->InstallEnbDevice (enbNodesPico.Get(2)));

    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandOffset", UintegerValue (42));
    lteHelper->SetFfrAlgorithmAttribute ("DlSubBandwidth", UintegerValue (8));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandOffset", UintegerValue (42));
    lteHelper->SetFfrAlgorithmAttribute ("UlSubBandwidth", UintegerValue (8));
    enbPicoDevs.Add(lteHelper->InstallEnbDevice (enbNodesPico.Get(3)));
 }
```

Jelikož mají různé typy eNodeB rozdílné vysílací výkony, tak musí být nastaven vysílací výkon pro makro a piko eNodeB zvlášť. K nastavení vysílacího výkonu slouží třída *LteEnbPhy,* respektive *LteUePhy.* Tyto třídy modelují fyzickou vrstvu daných uzlů. Ve třídách existují metody *SetTxPower,* které slouží pro nastavení vysílacího výkonu. Stejným způsobem nastavte i vysílací výkony všech UE v simulaci.

```
//Nastavení vysílacího výkonu pro jednotlivé eNB
Ptr<LteEnbPhy> macroEnbPhy = enbMacroDevs.Get (0)->GetObject<LteEnbNetDevice> ()->GetPhy
();
macroEnbPhy->SetTxPower (macroEnbTxPowerDbm);
… … …
```

Pro připojení UE k eNodeB se používá metoda *Attach* třídy *LteHelper.* Parametry metody jsou požadovaný UE a jemu příslušící eNodeB. Proveďte tento krok.

```
// Připojení UE k jim odpovídajícím eNB
lteHelper->Attach (ueMacroDevs, enbMacroDevs.Get (0));

for( uint16_t i = 0; i < numberOfPicoEnbs; i++ ){
  for (uint16_t j = 0; j < numberOfPicoUes/numberOfPicoEnbs; j++){
       lteHelper->Attach (uePicoDevs.Get (i*4+j), enbPicoDevs.Get (i));
    }
  }
```

Pomocí třídy *EpsBearer* nastavte parametry QoS na úrovni nosiče na *EpsBearer::GBR_CONV_VOICE*. Třída *EpsBearer* obsahuje specifikaci nosičů EPS. Pomocí metody *ActivateDataRadioBearer* třídy *LteHepler* aktivujte data radio bearer na všech UE v simulaci.

```
// Aktivace EPS nosiče
enum EpsBearer::Qci q = EpsBearer::GBR_CONV_VOICE;
EpsBearer bearer (q);
lteHelper->ActivateDataRadioBearer (ueMacroDevs, bearer);
lteHelper->ActivateDataRadioBearer (uePicoDevs, bearer);
```

Další části simulace je zapnutí trasování zdrojů (nahrávání statistik) a připojení trasovacích zdrojů pro sledované parametry, které nejsou obsaženy ve výchozím trasování zdrojů. Podle zadání protokolu by si měl čtenář vybrat dvě vhodné, rozdílné statistiky (trasovací zdroje) z první části protokolu a dvě rozdílné statistiky z druhé části protokolu a jejich průběhy vykreslit a popsat. Může se také inspirovat tímto návodem, ve kterém jsou vybrány jako dvě statistiky z první části tyto: *interference* a *rsrq,* a jako dvě vybrané statistiky z druhé části protokolu tyto: *propustnost piko buňky* a *SINR* (ty jsou součástí

funkcí *EnablePhyTraces* a *EnableRlcTraces*). Důležité je také zmínit fakt, že statistiky by se neměly začít nahrávat od úplného začátku simulace, ale až po tzv. zahřívací periodě, z důvodu možnosti zkreslení statistik simulace. Třída *RadioBearerStatsCalculator* slouží pro výpočet statistik *rlc* a *pdcp* vrstev. Statistiky se vypočítávají v po sobě jdoucích časových oknech a pravidelně se zapisují do souboru.

```
// Zapnutí nahrávání statistik
lteHelper->EnablePhyTraces ();
lteHelper->EnableMacTraces ();
lteHelper->EnableRlcTraces ();
lteHelper->EnablePdcpTraces ();
Ptr<RadioBearerStatsCalculator> rlcStats = lteHelper->GetRlcStats ();
rlcStats->SetAttribute ("StartTime", TimeValue (Seconds (0.3)));  //zahřívací perioda
rlcStats->SetAttribute ("EpochDuration", TimeValue (Seconds (simTime)));
Ptr<RadioBearerStatsCalculator> pdcpStats = lteHelper->GetPdcpStats ();
pdcpStats->SetAttribute ("StartTime", TimeValue (Seconds (0.3))); //zahřívací perioda
pdcpStats->SetAttribute ("EpochDuration", TimeValue (Seconds (simTime)));
```

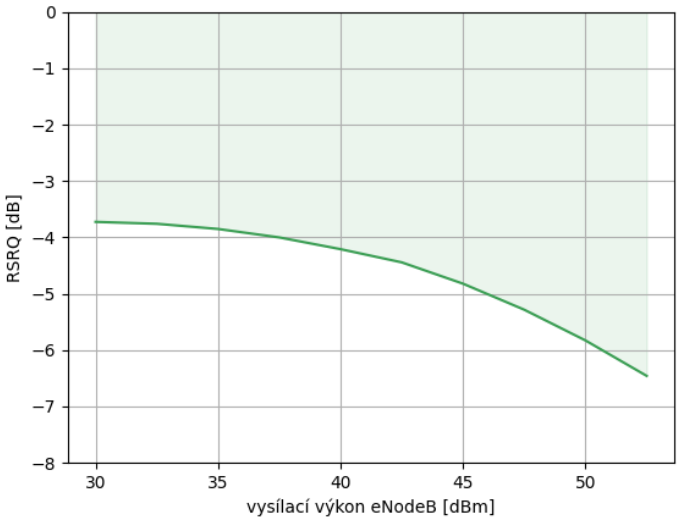
Metody *lteHelperu* s prefixem *Enable* a postfixem *Traces* vytváří v kořenovém adresáři simulace nové *.txt* soubory s výsledky simulace. Tyto soubory osahují veškeré základní statistiky, které jsou k nalezení v dokumentaci k těmto metodám (jedná se o statistiky jako přenosová rychlost, SINR apod.). Uživatel si poté může dané soubory zpracovat a pomocí vhodných nástrojů vykreslit výstupní grafy. Pro připojení trasovacích zdrojů pro zaznamenávání *rsrq* a *interference* do konfiguračního subsystému použijte tyto konfigurační cesty: */NodeList/*/DeviceList/*/ComponentCarrierMapUe/*/LteUePhy/ReportUeMeasurements*, */NodeList/*/DeviceList/*/$ns3::LteEnbNetDevice/ComponentCarrierMap/*/LteEnbPhy/ReportInterference.* Ukázka implementace callback metody pro libovolný trasovací zdroj a přidání trasovacího zdroje do konfiguračního subsystému je k dispozici v sekci 2.3.2. Metody pro zpětné volání se spouští v časových intervalech na základě odpovídajících třídních atributů s postfixem *SamplePeriod*. Je vhodné si na konci simulace vypočítat průměr ze všech měření z jednotlivých zpětných volání. Posledním volitelným krokem je konfigurace *NetAnim*. Pro jeho využití ve zdrojovém kódu je nutné přidat modul *ns3/netanim-module.h*. Ukázka konfigurace NetAnim je k nalezení v sekci 2.3.1.

3.4.5.2 *Výstup a zhodnocení výsledků laboratorního cvičení*

K zapnutí simulace se simulačním časem 30 sekund a vysílacím výkonem makro buňky 35 dBm lze použít tento příkaz:

```
./waf --run "simulationInterference --simTime=30 --macroEnbTxPowerDbm=35"
```

Pro vykreslení grafů byl opět použit programovací jazyk Python s knihovnami matplotlib, numpy, scipy a statsmodels. Cílem první části protokolu je popsat změny určitých parametrů v piko buňkách v závislosti na vysílacím výkonu makro buňky a ověřit tak, že student chápe souvislosti mezi jednotlivými výstupními statistikami. Následují jednotlivé grafy s teoretickým popisem a vysvětlením.

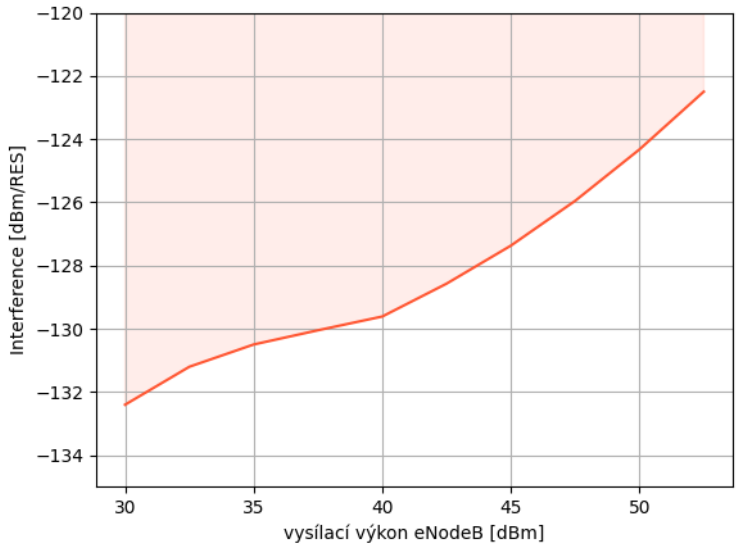


Obrázek 3.22: *Graf závislosti RSRQ na vysílacím výkonu makro eNodeB*

Prvním grafem je graf závislosti RSRQ (Reference Signal Received Quality- ukazatel kvality příjmu referenčního signálu) na vysílacím výkonu makro buňky. RSRQ se v LTE sítích používá jako ukazatel k určení kvality příjmu referenčního rádiového signálu. RSRQ na rozdíl od RSRP (Reference signal received power – přijatý výkon referenčního signálu) zahrnuje také úroveň rušení kvůli zahrnutí RSSI (Received signal strength indicator - ukazatel síly přijatého signálu) do výpočtu, což jde vidět na obrázku 3.22. RSSI se skládá ze síly signálu obsluhující buňky, sousední buňky ve společném kanálu, výkonem interference z ostatních buněk a šumu. Protože RSRQ je poměr dvou výkonů signálu se stejnou jednotkou (RSRP a RSSI, tj. dBm), také s ohledem na počet použitých resource bloků, tak RSRQ používá jako jednotku dB. Graf ukazuje, že s rostoucím vysílacím výkonem makro buňky dochází k poklesu RSRQ – dochází ke zhoršení ukazatele kvality příjmu referenčního signálu. Tento jev je způsoben z důvodu větší interference při vyšším vysílacím výkonu makro buňky. Nicméně v této simulaci hodnota RSRQ neklesne pod -7 dB, což lze stále považovat za více než dostačující hodnotu. Pro zjištění RSRQ posílá UE celočíselnou hodnotu eNodeB, která se podle mapovací tabulky přepočítává na odpovídající hodnotu v dB (tabulka 3.6).

Tabulka 3.6: *Mapovací tabulka RSRQ [32]*

Kód	Rozmezí RSRQ [dB]	Kód	Rozmezí RSRQ [dB]	Kód	Rozmezí RSRQ [dB]
0	<-19,5	12	-14,0 až -13,5	24	-8,00 až -7,50
1	-19,5 až -19,0	13	-13,5 až -13,0	25	-7,50 až -7,00
2	-19,0 až -18,5	14	-13,0 až -12,5	26	-7,00 až -6,50
3	-18,5 až -18,0	15	-12,5 až -12,0	27	-6,50 až -6,00
4	-18,0 až -17,5	16	-12,0 až -11,5	28	-6,00 až -5,50
5	-17,5 až -17,0	17	-11,5 až -11,0	29	-5,50 až -5,00
6	-17,0 až -16,5	18	-11,0 až -10,5	30	-5,00 až -4,50
7	-16,5 až -16,0	19	-10,5 až -10,0	31	-4,50 až -4,00
8	-16,0 až -15,5	20	-10,0 až -9,50	32	-4,00 až -3,50
9	-15,5 až -15,0	21	-9,50 až -9,00	33	-3,50 až -3,00
10	-15,0 až -14,5	22	-9,00 až -8,50	34	>-3,00
11	-14,5 až -14,0	23	-8,50 až -8,00		



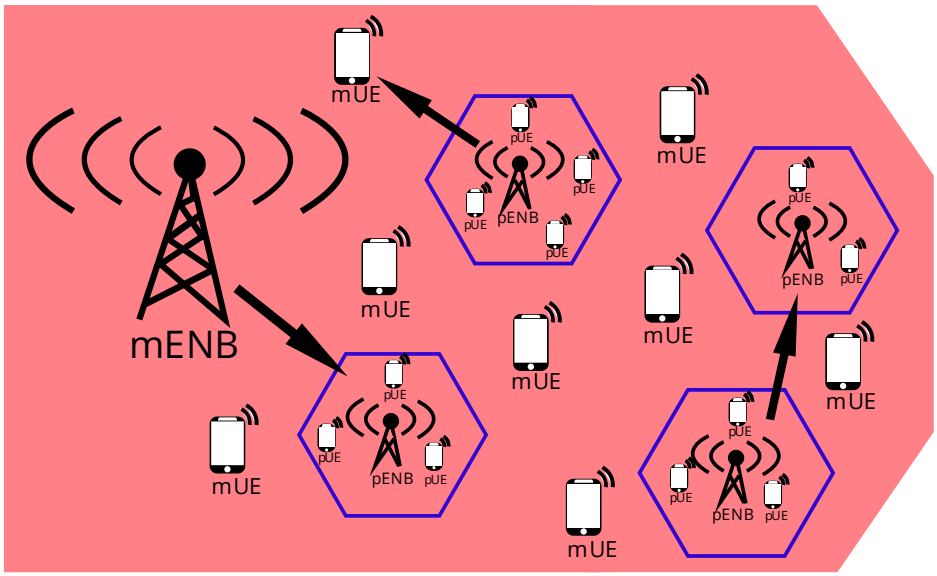
Obrázek 3.23: *Graf závislosti interference na vysílacím výkonu makro eNodeB*

Druhým grafem je graf závislosti interference na vysílacím výkonu makro eNodeB. V ns-3 simulátoru se pro zjištění interference používá trasovací zdroj *ReportInterference.* Tento trasovací zdroj vrací hodnotu výkonu interference včetně šumu v rámci jednoho resource bloku. Výstup je poté množina sestávající z přijatých výkonů interference v rámci každého resource bloku. Při sečtení těchto výkonů dostaneme výkon interference v rámci všech resource bloků (v této simulaci je jich 50). Obrázek 3.23 popisuje závislost interference na vysílacím výkonu makro buňky. Z grafu je patrné, že s rostoucím vysílacím výkonem makro buňky dochází k navýšení výkonu interference. Dochází ke zvětšení vlivů interference, což se projeví nižší hodnotou SINR, snížení výkonnosti sítě, zkušenosti uživatelů a snížení efektivity používání síťových zdrojů. Správné plánování frekvence tomu může do určité míry zabránit, nicméně interference nemůže být zcela eliminována. Přiblížení problematiky plánování frekvencí je cílem druhé části tohoto laboratorního cvičení.

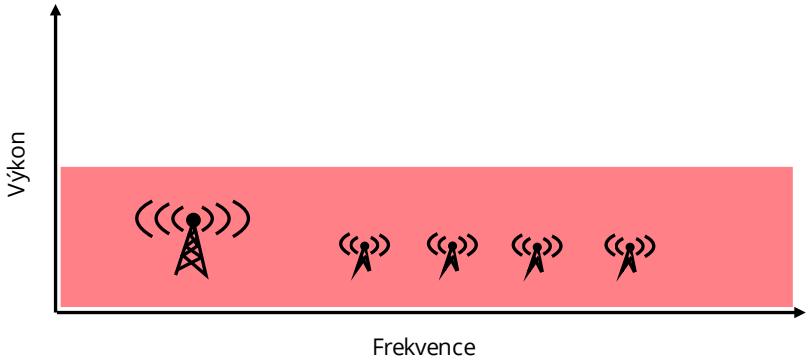
Pro zapnutí simulace pro potřeby druhé části cvičení lze použít příkaz níže. Jedná se o zapnutí simulace se simulačním časem 30 sekund, vysílací výkonovou úrovní makro buňky 46 dBm a typem použitého algoritmu je *LteFrNoOpAlgorithm.*

```
./waf --run "simulationInterference --simTime=30 --macroEnbTxPowerDbm=46
--frAlgorithm=NOOP"
```

Dalším cílem druhé části protokolu spolu k přiblížení problematiky plánování frekvencí je porovnání různých typů algoritmů pro opětovné použití frekvencí. Jedním z cílů studenta pro toto cvičení je pochopit souvislosti mezi jednotlivými výstupními statistikami v závislosti na použitém algoritmu. V této části jsou algoritmy čtenáři popsány blíže a pro úplnost jsou popsány všechny algoritmy pro opětovné použití frekvencí, které jsou uvedeny v zadání čtvrtého cvičení – tabulka 3.5. Po popisu jednotlivých algoritmů následují dva grafy statistik (výstup protokolu) s teoretickým popisem a vysvětlením (průměrná propustnost piko buňky a SINR).

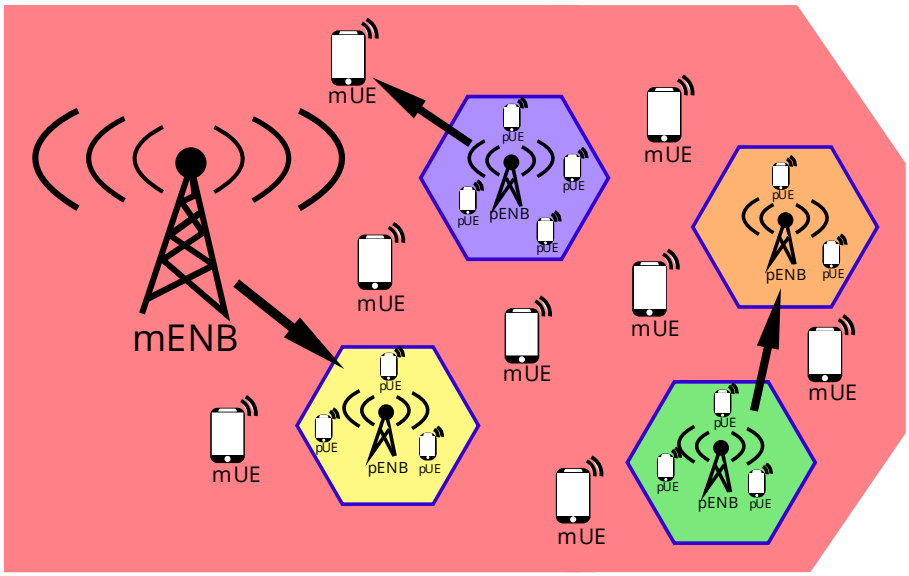


Obrázek 3.24a: *Architektura sítě při použití ns3::LteFrNoOpAlgorithm - žádné opakované použití frekvence*

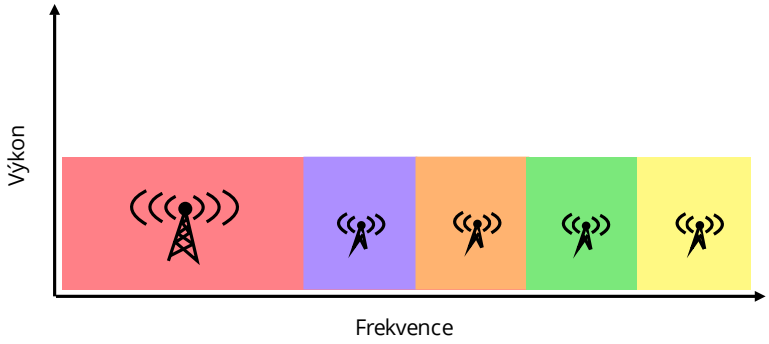


Obrázek 3.24b: *Využití frekvencí buňkami – NoOp algoritmus*

Schéma pro opětovné použití plné frekvence je implementováno pomocí algoritmu pro žádné opakované využití frekvencí. V navržené architektuře pro laboratorní cvičení se mezi eNodeB neprovádí žádné dělení frekvence, faktor opětovného použití frekvence se tedy rovná jedné. Makro a piko buňky v této síťové architektuře tedy využívají celou šířku pásma a vysílají stejnoměrným výkonem přes všechny resource bloky. Je to nejjednodušší schéma a základní způsob provozování sítě LTE. Výkon uživatelů v piko buňkách je omezen díky vysoké úrovni interference (jedna z nejvyšších z vybraných algoritmů), nicméně schéma umožňuje dosáhnout vysokých datových přenosových rychlostí. Díky tomu, že všechny buňky mohou vysílat na stejných frekvencích, schéma povoluje plánovači používat celou šířku pásma a umožňuje UE využívat jakýkoliv resource blok, dochází zde k vysoké úrovni interference. Pro implementaci tohoto algoritmu slouží třída *LteRfNoOpAlgorithm*, která jako všechny třídy pro opětovné použití frekvencí dědí z abstraktní třídy *LteFfrAlgorithm.* Tento algoritmus je v eNodeB v ns-3 simulátoru jako výchozí.

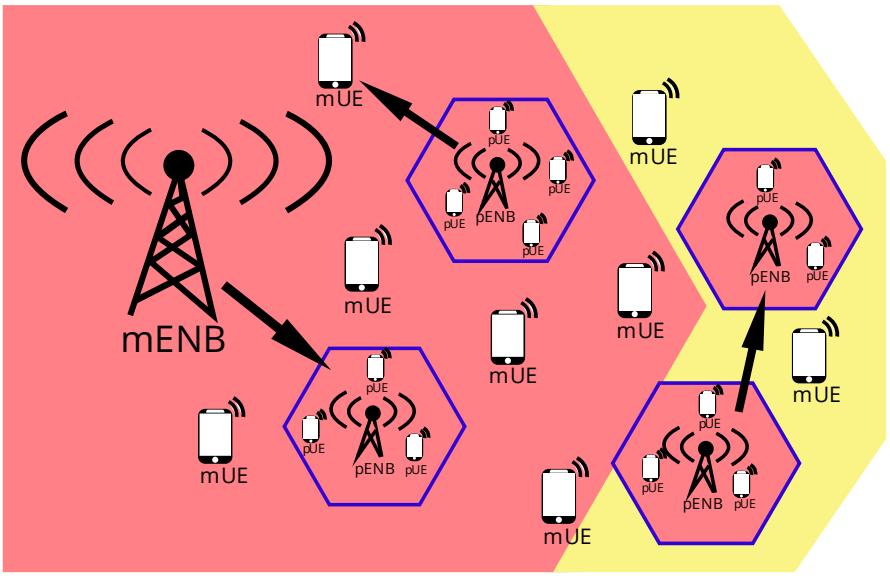


Obrázek 3.25a: *Architektura sítě při použití ns3::LteFrHardAlgorithm – „tvrdé“ opakované použití frekvence*

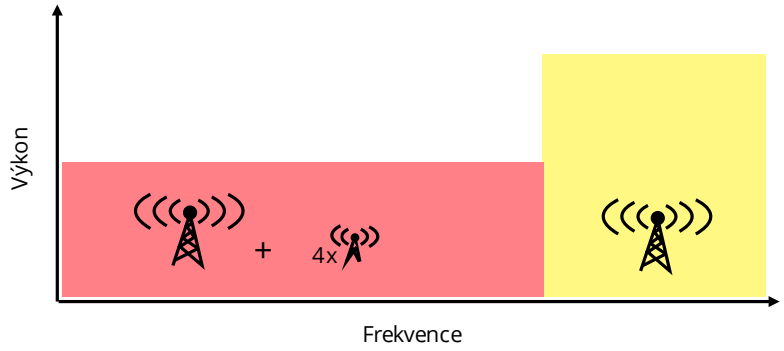


Obrázek 3.25b: *Využití frekvence buňkami – hard algoritmus*

V této síťové architektuře poskytuje algoritmus nejjednodušší schéma pro umožnění snížení úrovně interference mezi makro a piko buňkami. Celá šířka frekvenčního pásma je rozdělena na pět disjunktních dílčích pásem. Jednotlivým piko buňkám a makro buňce je přiděleno jiné subpásmo. Faktor opakovaného použití frekvence se rovná počtu dílčích pásem, tedy pěti. Toto schéma umožňuje výrazně snížit interferenci a zlepšit zkušenost uživatelů. Vzhledem k tomu, že každý eNodeB využívá pouze jednu část celé šířky pásma, je úroveň datové rychlosti snížena v průměru o faktor rovný faktoru opětovného použití frekvence. V této síťové architektuře dosahuje algoritmus ze všech algoritmů implementovaných v laboratorním cvičení nejmenších hodnot interference, za cenu nejmenších hodnot přenosových rychlostí. Pro implementaci algoritmu slouží třída *ns3::LteFrHardAlgorithm.*

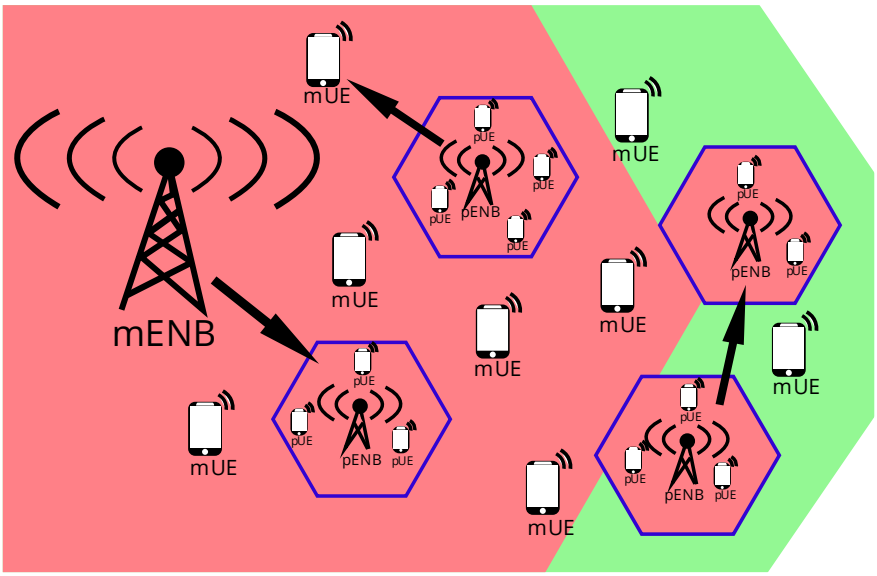


Obrázek 3.26a: *Architektura sítě při použití ns3::LteFrStrictAlgorithm – „striktní“ opakované použití frekvence*

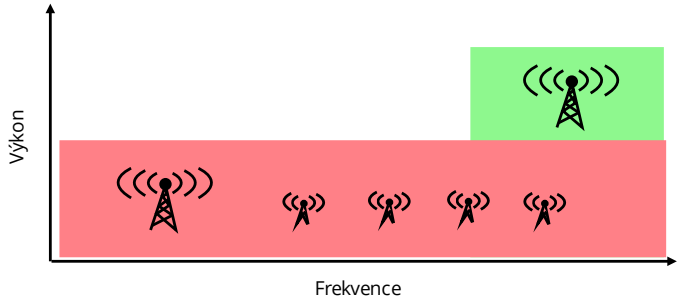


Obrázek 3.26b: *Využití frekvence buňkami – strict algoritmus*

Výše uvedený algoritmus je kombinací předchozích dvou algoritmů pro znovupoužití frekvence. Spočívá v rozdělení frekvenčního pásma na dvě části s rozdílnými bloky pro znovupoužití frekvence. Jedno společné pásmo je používáno uvnitř všech makro a piko buněk, zatímco zbylá část je používána pouze makro buňkou a vysílá se v ní vyšším výkonem. Algoritmus původně počítá s více rovnocennými sousedními buňkami (například makro – makro), nicméně pro potřeby této simulace byl nastaven tak, jak je zobrazen na obrázku 3.26a. Účelem je vytvoření jednoho dílčího pásma s nízkou úrovní mezibuněčného rušení. Makro UE ve středu buňky, stejně tak i piko buňky a jejich UE, mají plně znovu použitelné frekvenční bloky, zatímco pro UE na okrajích makro buňky jsou vyhrazeny odlišné bloky, které mají minimální úroveň interference s ostatními částmi sítě. To znamená, že vnitřní UE piko buněk nesdílí žádné spektrum s okrajovými UE makro buňky, což snižuje interferenci pro uzly sítě. Hranice mezi pásmy lze nastavit pomocí prahové hodnoty RSRQ, což je parametr třídy *ns3::LteFrStrictAlgorithm.*

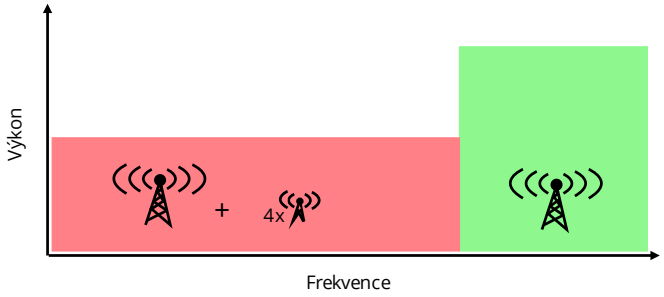


Obrázek 3.27a: *Architektura sítě při použití ns3::LteFrSoftAlgorithm – „měkké“ opakované použití frekvence*



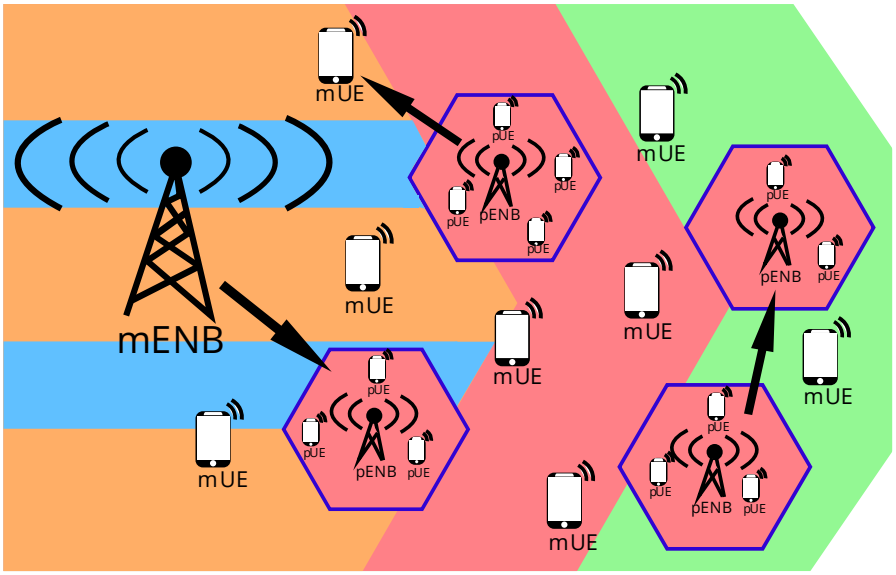
Obrázek 3.27b: *Využití frekvence buňkami – soft1 algoritmus*

Zde existují dvě sub-pásma, ve kterých jsou UE obsluhovány jinou výkonovou úrovní, navíc při tomto algoritmu všechny eNodeB vysílají v celé šířce pásma. V síťové architektuře všechny UE v piko buňkách sdílí stejnou šířku pásma s centrálním pásmem makro buňky, vysílají na nižší výkonové úrovni v porovnání s UE na kraji makro buňky. Soft algoritmus je efektivnější z hlediska šířky pásma oproti striktnímu algoritmu, protože využívá celou šířku pásma systému, nicméně tento algoritmus také vede k většímu rušení jak uvnitř buněk, tak na jejich okrajích. Algoritmus je rozdělen na dva typy – v této práci jsou označeny jako *soft1* a *soft2*. V první verzi je frekvenční pásmo rozděleno na dvě subpásma s možností vysílat s nižším a vyšším vysílacím výkonem. Centrální subpásmo (subpásmo s možností nižšího výkonu) je přístupné pouze pro centrální UE. Druhé subpásmo, které je vyhrazené pro krajní UE, může být využíváno i centrálními UE, ale pouze s redukovaným výkonem a v případě, že není obsazeno uživateli, kteří se nacházejí na okraji buňky. Druhá verze je popsána na obrázku níže.

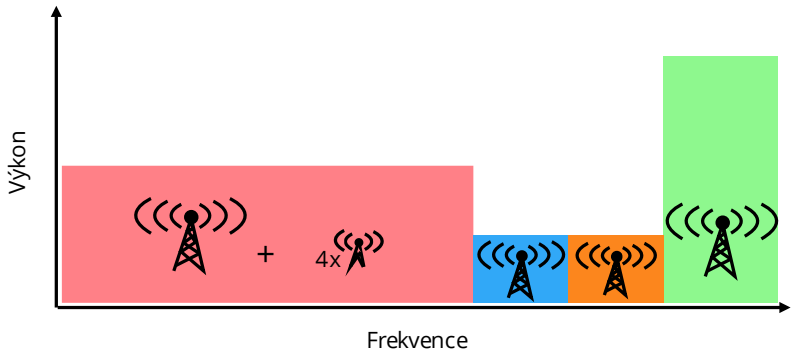


Obrázek 3.27c: *Využití frekvence buňkami – soft2 algoritmus*

Ve druhé verzi nemají centrální UE přístup ke krajnímu subpásmu. V architektuře, která je použita pro laboratorní cvičení je tento druhý typ algoritmu totožný jako v případě použití striktního algoritmu. V takovém případě mohou piko buňky využívat pouze centrální subpásmo a krajní subpásmo je vyhrazeno makro buňce, čímž dochází k redukci interference. To ovšem znamená, že nižší hodnota interference na okrajích makro buňky je dosažena za cenu nižšího využití spektra. K rozhodnutí, s jakou úrovní výkonu by se mělo UE obsluhovat, využívá algoritmus měření UE a porovnává je s prahovou hodnotou atributu RSRQ.



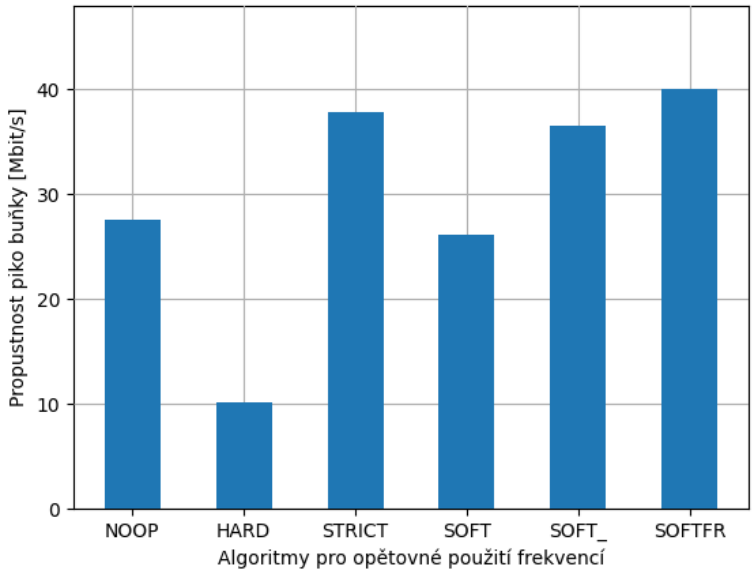
Obrázek 3.28a: *Architektura sítě při použití ns3::LteFfrSoftAlgorithm – „měkké frakční“ opakované použití frekvence*



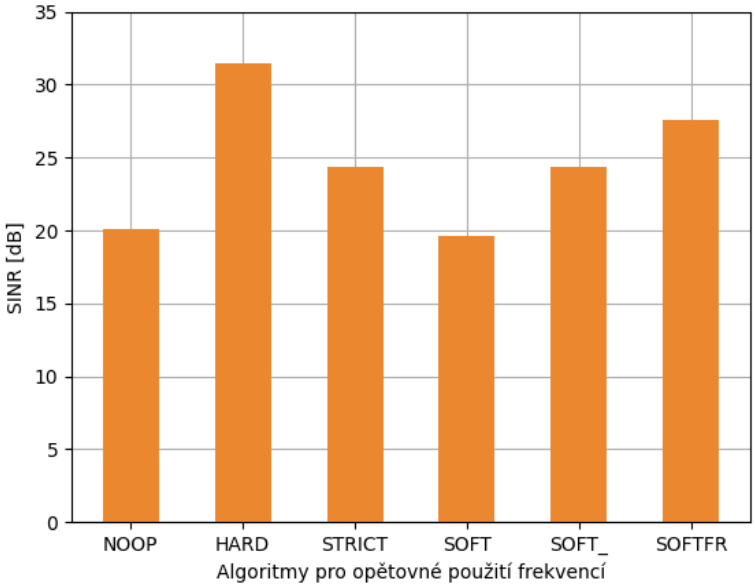
Obrázek 3.28b: *Využití frekvence buňkami – soft fractional algoritmus*

Měkký frakční algoritmus využívá dílčí pásma přidělená pro okraj jiných buněk pro vnitřní UE s nízkým vysílacím výkonem, tím se liší v porovnání se striktním algoritmem, jelikož striktní algoritmus nepoužívá subpásma přidělená pro vnější oblasti. V síťové architektuře je algoritmus využíván pouze makro buňkou, s tím že každá piko buňka využívá největší společnou část rozděleného frekvenčního spektra (obrázek 3.28b). Díky tomu používá měkký frakční algoritmus stejně jako klasický měkký algoritmus subpásmo s vysokou úrovní vysílacího výkonu a s nízkou úrovní vysílacího výkonu, navíc využívá dvě subpásma s nižším vysílacím výkonem v centru makro buňky za účelem znatelnějšího snížení interference a možnosti zvýšení propustnosti vnitřních uživatelům. Tím se liší oproti měkkému

algoritmu. Díky těmto odlišnostem dosahuje algoritmus menších hodnot mezibuněčného rušení a za cíl si klade maximalizaci přenosových rychlostí.



Obrázek 3.29: *Srovnání průměrné propustnosti v piko buňce v závislosti na použitém typu algoritmu*



Obrázek 3.30: *Srovnání SINR jednotlivých algoritmů pro opětovné použití frekvencí*

Graf 3.29 ukazuje srovnání průměrné propustnosti piko buňky v závislosti na použitém typu algoritmu. Význam jednotlivých zkratek algoritmů na ose x je následující: NOOP- *ns3::LteFrNoOpAlgorithm,* HARD - *ns3::LteFrHardAlgorithm,* STRICT - *ns3::LteFrStrIctAlgorithm,* SOFT - n*s3::LteFrSoftAlgorithm* (povolený přístup centrálních UE ke krajnímu subpásmu), SOFT_ - *ns3::LteFrSoftAlgorithm* (přístup centrálních UE ke krajnímu subpásmu není povolen), SOFTFR - n*s3::LteFfrSoftAlgorithm.* Z grafu je patrné, že nejmenší propustnost je dosažena při použití *HARD* algoritmu. Tento algoritmus má ze všech použitých algoritmů největší faktor opakovaného použití frekvence, ten se rovná počtu subpásem. Celá šířka pásma je rozdělena na pět disjunktních subpásem. Každá piko buňka může využívat jen malou část celé šířky pásma, což se projevuje na nejmenší propustnosti. Nicméně rozdělení na disjunktní dílčí pásma má za následek výrazné snížení interference, interference je velmi malá – nejmenší ze všech typů srovnávaných algoritmů, což je vidět

na obrázku 3.30. Hodnota SINR u *HARD* algoritmu je největší ze všech srovnávaných algoritmů v důsledku právě oné nejmenší interference. Algoritmy *NOOP* a *SOFT* jsou téměř identické z hlediska srovnávaných parametrů. Oba algoritmy mají největší hodnoty interference ze všech typů algoritmů a druhou nejmenší propustnost. Je to způsobeno tím, že makro a piko buňky v síťové architektuře při použití těchto algoritmů využívají celou šířku pásma. Výkon uživatelů je v buňkách omezen z důsledku vysoké úrovně interference. V případě *SOFT* vysílají uživatelé na okraji buňky větším výkonem a používají část stejného pásma spolu s piko buňkami a makro buňkou. To má za následek menší hodnoty propustnosti a menší hodnoty SINR v porovnání s *NOOP* algoritmem.

U algoritmů *STRICT* a *SOFT_* je frekvenční pásmo rozděleno na dvě disjunktní dílčí pásma s cílem umístit dvě piko buňky do oblasti s použitím druhého dílčího pásma. Jak už bylo zmíněno, tak v této architektuře mají algoritmy stejné vlastnosti, jelikož *SOFT_* algoritmus se používá pouze pro jednu buňku – makro. Kdyby byl použit pro více rovnocenných sousedních buněk, pak by došlo k rozdělení druhého subpásma na další disjunktní pásma podle počtu daných buněk. UE na krajích mají vyhrazenou část pásma pro sebe, kde mají minimální úroveň rušení s ostatními částmi sítě. To má za následek větší propustnost a větší hodnoty SINR (obrázky 3.29 a 3.30). Posledním srovnávaných algoritmem je *SOFTFR.* Tento algoritmus dosahuje největší propustnosti a druhé nejvyšší hodnoty SINR. Algoritmus používá dílčí pásma i pro centrální UE s nízkým vysílacím výkonem. Jelikož každá piko buňka využívá menší společnou část rozděleného spektra, dochází k menším hodnotám interference. Rozdělení na větší počet subpásem spolu s vysokou, nízkou úrovní vysílacího výkonu a použití společného dílčího pásma vede ke zvýšení propustnosti a snížení vlivu interference v piko buňkách.

4 Srovnání nástrojů SimuLTE a ns-3

Cílem kapitoly je srovnat oba nástroje z pohledu možností, výkonnosti, efektivity práce s nástroji, náročnosti a rozsahu funkcí. Pro potřeby porovnání výkonnosti byl implementován v obou nástrojích stejný, rozsáhlý simulační scénář s možností nastavit libovolný počet eNodeB a libovolné množství generovaného provozu v síti. Pro potřeby dalšího porovnání byly implementovány simulační příklady zabývající se podobnou problematikou, které využívají oba moduly, ty jsou popsány v předchozí kapitole.

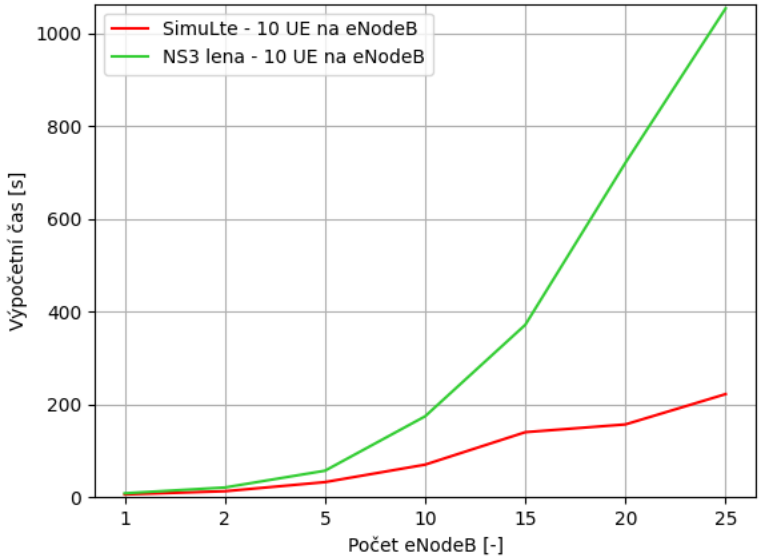
4.1 Porovnání výkonnosti

Simulační scénáře pro porovnání výkonnosti jsou k dispozici v elektronických přílohách C a E. Simulační scénáře byly testovány v systému Ubuntu 18.04 LTS, jako procesor byl použit Intel Core i5-6500 spolu s 8 GB RAM. Modul LTE byl sestaven v optimalizovaném režimu. Konfigurace LTE modulu byla následující: *./waf -d optimized --enable-examples --enable-tests configure*. Pro potřeby SimuLTE frameworku byl použit textový režim v prostředí *cmdenv*. Ten byl nastaven v režimu verbose, v expresním a interaktivní módu. Pro zápis výsledných simulačních dat byl použit skript napsaný v pythonu, který je k nalezení v elektronické příloze H. Tento skript měří i čas doby běhu simulace, respektive procesu, nicméně pro potřeby měření času byly použity interní funkce, které daný modul/framework poskytuje. Pro běh python skriptu je potřeba nainstalovat knihovny pandas a psutil. Skript potřebuje dva vstupní parametry na příkazovém řádku, název procesu, který má monitorovat a jméno výstupního souboru. Výsledky ukládá do souboru typu CSV. Pro vykreslení grafů byl použit python spolu s knihovnami matplotlib a numpy.

V simulačních scénářích je možnost nastavit počet eNodeB, čas mezi přenosem paketů, velikost přenášených paketů a dobu simulace. Simulační scénáře jsou napsány tak, že vytvářejí pomyslnou mříž složenou z eNodeB vždy po maximálně pěti eNodeB na jeden řádek. Počet řádků je dán zcela na zvoleném celkovém počtu eNodeB. Ke každému eNodeB je vždy pevně přiděleno 10 UE, celkem se tedy s každým eNodeB v simulaci navíc přidá 11 uzlů. Vzdálenost mezi jednotlivými eNodeB je 500 metrů. UE jsou rozmístěny náhodně v rozmezí 300 metrů od eNodeB. Simulační scénáře vytvářejí síť EPC a externího hostitele připojeného k ní prostřednictvím internetu. Každý UE odesílá a přijímá data do a ze vzdáleného hostitele. Každá monitorovaná simulace byla provedena celkem desetkrát s tím, že výsledky byly zprůměrovány. Následuje stručná tabulka srovnání výkonnosti frameworků z provedených simulací.

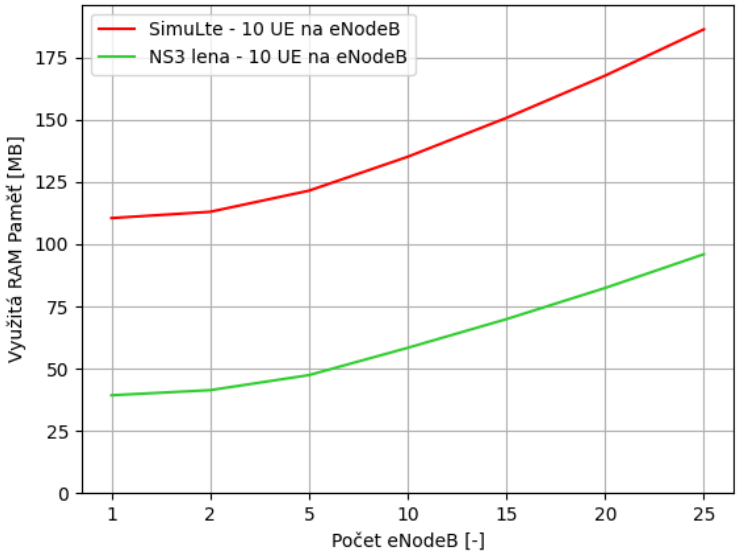
Tabulka 4.1: *Srovnání výkonnosti*

	ns-3 LTE	**SimuLTE**
Využití paměti	**nižší**	**vyšší**
CPU zátěž	**shodná**	
Doba běhu simulace	**vyšší**	**nižší**



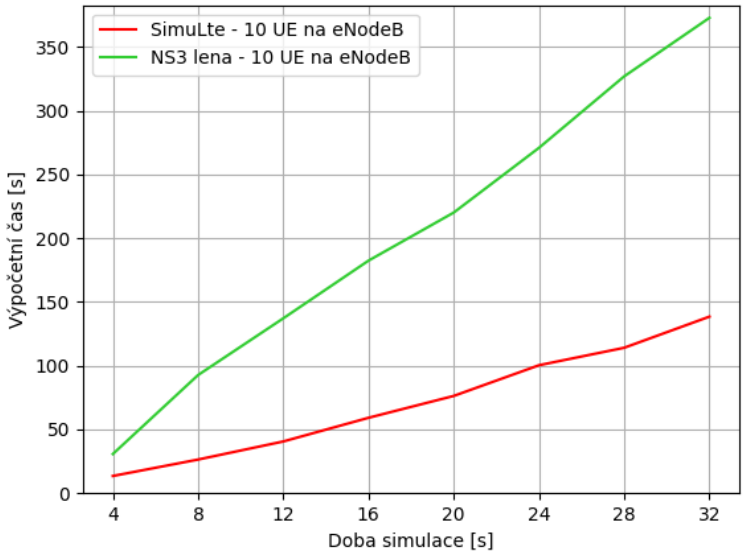
Obrázek 4.1: *Graf závislosti výpočetního času simulace na počtu eNodeB a uzlů v síti– doba simulace 20 sekund*

Výše uvedený graf vyhodnocuje výpočetní čas simulace (reálnou dobu běhu) pro pevnou dobu simulace 20 sekund v závislosti na počtu eNodeB v simulačním scénáři. Obrázek ukazuje očekávané chování, kdy s rostoucím počtem uzlů v simulaci roste reálná výpočetní doba. Lze si všimnout, že při větším počtu eNodeB dochází ke značnému rozdílu ve výpočetních dobách jednotlivých simulátorů. Tento rozdíl je větší, čím větší počet uzlů v simulaci je použit.



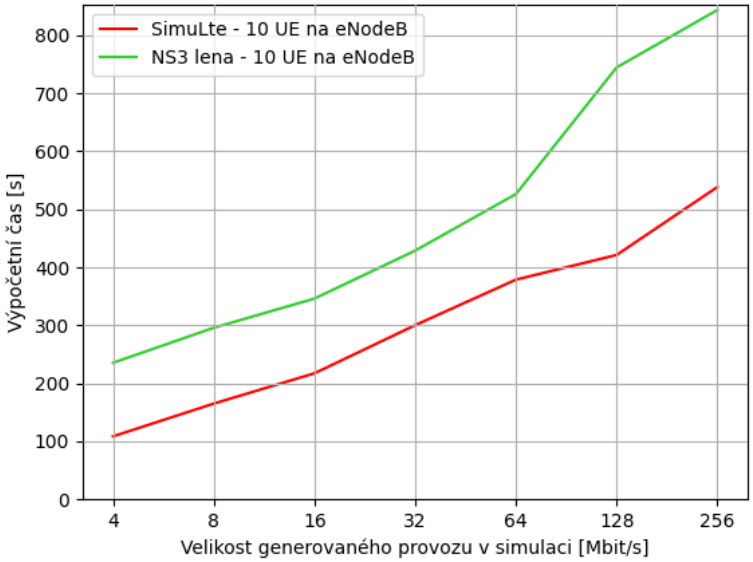
Obrázek 4.2*: Graf závislosti využití RAM paměti na počtu eNodeB a uzlů v síti – doba simulace 20 sekund*

Obrázek 4.2 vyhodnocuje využití operační paměti pro pevnou dobu simulace 20 sekund v závislosti na počtu eNodeB v simulaci. Očekávaně s rostoucím počtem eNodeB dochází k většímu využití operační paměti. Ns-3 modul LTE vykazuje daleko větší výpočetní doby při větším počtu eNodeB, nicméně využívá v průměru o méně než polovinu operační paměti oproti SimuLTE a cmdenv prostředí.



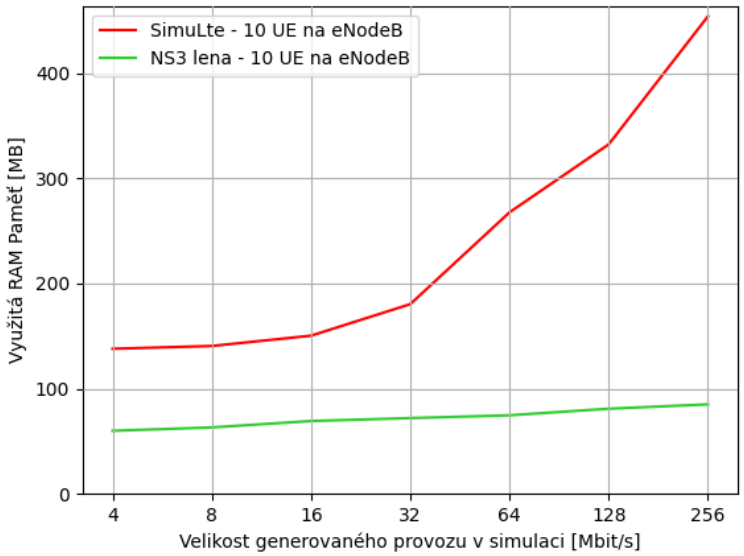
Obrázek 4.3: *Graf závislosti doby simulace na výpočetní čase při stálém počtu 10 eNodeB*

Na obrázku 4.3 je ověřena linearita výpočetní doby v závislosti na době simulace v obou nástrojích. Při pohledu na obrázek lze považovat obě funkce za lineární. Výpočetní čas v závislosti na době simulace je větší v LTE modulu ns-3 simulátoru. Počet eNodeB byl pro tento scénář pevně nastaven na 10. V předchozích třech scénářích byl generován VOIP provoz s velikostí paketů 100 bajtů a časem mezi přenosem paketů 100 milisekund.



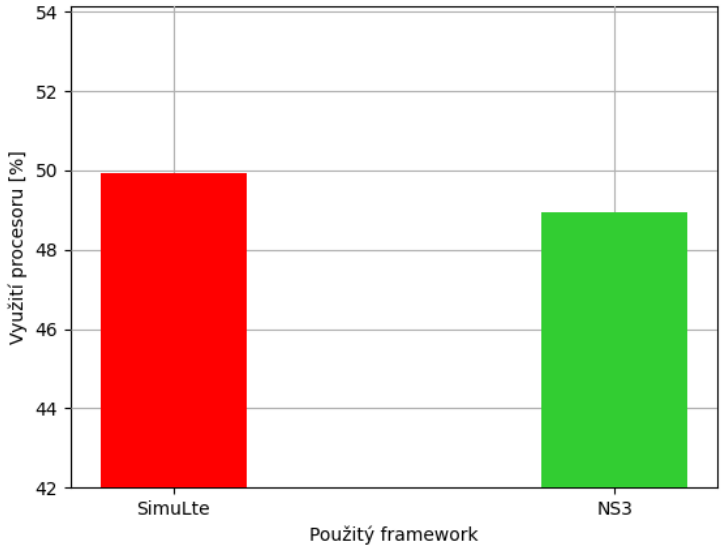
Obrázek 4.4: *Graf závislosti výpočetní doby na velikosti generovaného provozu v simulaci za sekundu – 10 eNodeB, doba simulace 20 sekund*

Na obrázku 4.4 je vyhodnocení výpočetního času simulace v závislosti na velikosti generovaného provozu v simulačním scénáři za sekundu. Simulace má nastavenou pevnou simulační dobu 20 sekund a pevný počet eNodeB, který je nastaven na 10. Podle očekávaného chování ukazuje obrázek závislost, kdy s rostoucí velikosti generovaného provozu v simulaci roste skutečný výpočetní čas. Lze si všimnout, že zobrazené závislosti na obrázku rostou lineárně - čím větší je velikost generovaného provozu za sekundu, tím větší je celkový výpočetní čas simulace.



Obrázek 4.5: *Graf závislosti využití operační paměti na velikosti generovaného provozu v simulaci za sekundu – 10 eNodeB, doba simulace 20 sekund*

Na obrázku 4.5 je ukázka vyhodnocení využití operační paměti v závislosti na velikosti generovaného provozu v simulačním scénáři za sekundu. Simulace má znovu nastavenou pevnou simulační dobu 20 sekund a pevný počet eNodeB na 10. Předpokládaně s rostoucí velikostí generovaného provozu v simulaci roste velikost využití operační paměti. Z obrázku je patrné, že u větších velikostí generovaného provozu v simulačním scénáři, v důsledku zdvojnásobení velikosti generovaného provozu, roste využití operační paměti SimuLTE frameworku více než desetkrát rychleji oproti ns-3. Ns-3 modul LTE má růst lineární.



Obrázek 4.6: *Graf závislosti průměrného využití procesoru na použitém frameworku*

Obrázek 4.6 popisuje průměrné využití procesoru obou frameworků v monitorovaných simulačních scénářích. Využití procesoru obou frameworků se liší v jednotce procenta, tudíž můžeme konstatovat, že využití procesoru lze u obou frameworků považovat za shodné.

4.2 Porovnání z hlediska možností, efektivity a rozsahu funkcí

Mezi jednoznačné výhody ns-3 frameworku a modulu LTE patří možnost si zobrazit a editovat veškeré zdrojové kódy, modely a simulátory. Nové modely mohou být tedy napsány přímo uživatelem. Pro práci s modulem i simulátorem je nutná znalost programování v C++. Stejné výhody platí i pro SimuLTE framework. SimuLTE je také zcela napsán v C++ a je plně přizpůsobitelný. Uživatel může vytvářet nové moduly, protokoly a implementovat nové algoritmy. Ukázka vytvoření vlastní statistiky a její doimplementování přímo do zdrojového kódu simulátoru je k dispozici v sekci 3.1.5.2. Jedná se také o otevřený projekt. Propůjčuje si koncept modularity od OMNeT++, to znamená, že je jednoduchý na rozšíření.

Nevýhodou SimuLTE frameworku je limitovaná dokumentace testů a celková implementace testů. Jako testy jednotlivých komponent a funkcí frameworku jsou použity pouze simulace uložené v části příklady (určené pro příklady uživatelům). Další nevýhodou je velice stručná a limitovaná dokumentace, která není v porovnání s dokumentací LTE modulu zdaleka tak podrobně zpracována a strukturována. Vyjma odborných publikací je dokumentace vedena jako jedna velice stručná webová stránka. Nicméně vytváření simulací je z určité míry založeno na OMNeT++ a INET frameworku, a ty jsou dokumentovány v daleko větším a dostačujícím rozsahu. V případě ns-3 LTE modulu je dokumentována každá komponenta systému, veškeré její funkce, a navíc jsou přidány i příklady v podobě testů. Dokumentace celého LTE modulu i ns-3 samotného je značně rozsáhlá, což hodnotím jako velké plus. Podařilo se mi i narazit na zavádějící informace v dokumentaci při vytváření ukázkových příkladů v předchozí kapitole, a to při rozdělení resource bloků u algoritmů pro opětovné použití frekvencí a prahové hodnoty RSRQ. Ty jsou v nejnovější verzi modulu již odstraněny. SimuLTE a LTE modul mají flexibilní architekturu, která umožňuje rychlé a snadné konfigurace síťových zařízení. Oba mohou být přizpůsobené, uživatelé mohou implementovat jakoukoliv požadovanou funkci.

Grafická vizualizace je podstatnou částí síťových simulátorů, která umožňuje vývojářům pochopit velké množství dat produkovaných během simulace sítě a ověřování vstupů. Ns-3 LTE modul je možné vizualizovat pomocí animátoru NetAnim. Nicméně slouží pouze pro vizualizací až výsledné simulace sítě po jejím provedení. SimuLTE spolu s OMNeT++ má pokročilé GUI s inteligentní podporou. Vizualizační modul je oddělen od simulátoru a je interaktivní s uživatelem přímo v průběhu simulace, což umožňuje reagovat na určité stavy simulace. Podporuje 2D i 3D režim a navíc mezi jeho velkou výhodu patří možnost zobrazení statistik simulace, ostatních proměnných, stavů a atributů uzlů za běhu simulace, což uživatel převážně využije v případě ladění. Tento grafický vizualizér umožňuje i případnou manipulaci a vykreslení statistik do grafů po skončení simulace. To jsou jednoznačné výhody oproti omezenému NetAnimu. Z pohledu rychlosti učení je SimuLTE efektivnější, dle mého názoru má daleko strmější křivku učení než modul LTE. Efektivita GUI u SimuLTE spolu s OMNeT++ je excelentní.

4.2.1 Možnosti a funkce frameworků

Co se týká srovnání frameworků z hlediska možností, tak SimuLTE podporuje celkem pět aplikací pro generování provozu (konstantní bitový provoz, voip apod.). Ns-3 LTE modul nabízí větší počet aplikací, s tím že má navíc oproti SimuLTE třídu pro EPS nosiče a výčtový typ pro indikaci třídy QoS podle specifikace 3GPP 23.203 sekce 6.1.7.2. [31] SimuLTE implementuje celkem šest propagačních modelů, u kterých je možné nastavit počet rádiových kanálů, frekvenci, ztrátový koeficient, útlum a maximální výkon. Ns-3 modul LTE na rozdíl od SimuLTE neimplementuje žádný propagační model a zcela spoléhá

na samotný oddělený propagační modul, který implementuje celkem 20 rozdílných propagačních modelů. Ns-3 propagační modul definuje dvě obecná rozhraní pro modelování ztrát a zpoždění signálu. Model lze zřetězit k jinému a vytvořit seznam. Tímto je možné dosáhnout modelu pomalého a rychlého slábnutí signálu nebo samostatně modelovat různé efekty oslabení.

SimuLTE nabízí celkem pět předefinovaných scénářů, podle kterých počítá útlum, jmenovitě – vnitřní hotspot, městská mikro buňka, městská makro buňka, venkovská makro buňka a předměstská makro buňka. Ns-3 modul LTE žádné předdefinované scénáře pro výpočet útlumu nenabízí. Uživatel si pomocí dostupných modulů pro propagaci, budov apod. musí vytvořit scénář samostatně. Oba moduly implementují handover založený na X2 rozhraní. SimuLTE implementuje pouze jeden handover algoritmus založený na prahové hodnotě RSSI. U tohoto algoritmu je možnost nastavit zpoždění handoveru. Modul LTE nabízí celkem tři algoritmy pro handover. První implementace handover algoritmu nedělá vůbec nic. Zvolení tohoto algoritmu je ekvivalentní k vypnutí automatického hadnoveru. Je to výchozí možnost modulu a slouží pro možnosti manuálních handoverů. Další dva algoritmy slouží pro automatický handover na základě hodnoty RSRP, kde je možné nastavit hodnotu času spuštění a hysterezi. Poslední algoritmus je založen na základě měření RSRQ hodnoty, kdy dojde k automatickému přepnutí, pokud je hodnota RSRQ obsluhující buňky menší než prahová hodnota (obdobná funkce je i u předchozího algoritmu). Na rozdíl od SimuLTE, ns-3 LTE modul podporuje indikaci chyby rádiového spojení na základě nízké úrovně signálu, ovšem zde chybí implementace selhání přepojení v důsledku selhání rádiového spojení, což například v rozšiřujících modulech LTE jako je ELENA (extension for ns-3 LTE module) implementováno již je.

SimuLTE navíc podporuje sítí řízenou komunikaci zařízení k zařízení. Hlavní funkce tohoto rozšíření v jsou jednotlivá přímá komunikace mezi UE, vylepšení modulů vrstev pro manipulaci zpráv, specifikace možností rozdílných vysílacích výkonů pro uplink a downlink komunikaci, a především podpora komunikace jeden k mnoha. Ns-3 modul LTE možnost komunikace zařízení k zařízení neimplementuje. SimuLTE implementuje model pro CoMP, který se řídí paradigmatem master-slave, kde hlavní uzel přijímá koordinační rozhodnutí a spoléhá na informace zaslané ze slave uzlů prostřednictvím X2 rozhraní. Za tímto účelem je každý eNodeB vybaven modulem, jmenovitě LteCompManager, který provádí operace související s CoMP a bezproblémově spolupracuje s rozhraním X2. Modul LTE v ns-3 simulátoru CoMP problematiku neimplementuje. Ns-3 implementuje algoritmy pro znovupoužití frekvencí, které byly srovnány v posledním návodu laboratorních cvičení. Ns-3 navíc díky potřebě určitých algoritmů pro znovu použití frekvencí podporuje implementaci kontroly výkonové úrovně. Implementuje kontrolu výkonové úrovně v obou směrech provozu, uplink i downlink. Podporuje dva režimy otevřená a uzavřená smyčka. SimuLTE režimy kontroly výkonu nepodporuje.

Pro potřeby agregace nosných komponent existuje v ns-3 LTE pomocná třída, která danou problematiku konfiguruje (CcHelper). Aktuálně je implementována agregace nosných komponent jak v režimu downlink, tak v režimu uplink. V SimuLTE si musí uživatel vytvořit EPC z dostupných uzlů samostatně, zatímco v ns-3 LTE existuje pomocná třída, která vytvoří EPC, nakonfiguruje EPC automaticky a podporuje dva režimy, kdy v případě prvního režimu se sloučí funkcionalita PGW a SGW do jednoho uzlu. SimuLTE plánuje do budoucna dodělat použití baterií v rámci UE, zatím tato funkčnost není implementována.

Závěr

Cílem diplomové práce bylo popsat vlastnosti SimuLTE frameworku pro simulační nástroj OMNet++, popsat vlastnosti modulu LTE pro simulační nástroj ns-3, nástroje srovnat a navrhnout úlohy pro odborný předmět. V teoretické části jsem se věnoval popisem vlastností SimuLTE, framework jsem popsal z hlediska architektury a modulů, které odpovídají vrstvám LTE protokolového modelu. Dále jsem krátce popsal simulační nástroj OMNeT++ spolu s frameworkem INET, na kterém je založena funkčnost velkého množství dalších simulačních frameworků, mezi které spadá i mnou popisovaný a využívaný SimuLTE. Framework jsem popsal i z hlediska vytváření simulačních scénářů, postupu měření statistik a možnosti zajištění náhodných a opakovatelných běhů simulací. Ve druhé teoretické části jsem se zaměřil na popis modulu LTE. Při popisu jsem stejně jako v případě SimuLTE kladl důraz na architekturu a části modulu, které odpovídají vrstvám LTE protokolového modelu. Následně jsem popsal možnosti vývoje simulačních scénářů v prostředí ns-3, různé nástroje pro grafické reprezentace výstupů simulací, které tvoří jednu z nejdůležitějších součástí simulací, dále pak způsob nahrávání a připojení trasovacích zdrojů do konfiguračního subsystému. Závěry obou teoretických kapitol jsem doplnil o stručný postup instalace frameworků a pro ně potřebných simulačních nástrojů.

V praktické části diplomové práce jsem se zabýval návrhem a realizací simulačních příkladů, které využívají oba moduly. Příklady se zabývají podobnou problematikou. Cílem příkladů je srovnání obou nástrojů z pohledu nabízených možností v problematice, kterou se navržené příklady zabývají. Navržené příklady jsem rozdělil na pět podkapitol, zabývající se cílem laboratorního cvičení, zadáním, architekturou sítě, parametry simulace a postupem řešení spolu s vyhodnocením dosažených výsledků simulačního scénáře. Příklady jsem zaměřil na specifické vlastnosti nástrojů v dané problematice. Scénáře jsou navrženy takových způsobem, aby všechny podkapitoly dohromady poskytovaly čtenáři kompletní postup pro vypracování příkladů. První simulační scénář se zabývá problematikou handoveru v LTE sítích. Jeho cílem je demonstrace funkcionalit SimuLTE frameworku a modulu LTE v rámci problematiky handoveru implementované v nástrojích a pochopení změny určitých parametrů v závislosti na rychlosti UE v LTE síti. Druhý simulační scénář se zabývá problematikou interference ve dvouvrstvé LTE sítí. Cílem tohoto simulačního scénáře je demonstrace funkcionalit SimuLTE frameworku a modulu LTE v rámci problematiky interference. Ukázka a pochopení problému interference v piko buňkách v závislosti na vysílacím výkonu makro buňky s tím, že v každém nástroji je poté příklad doplněn o další možnosti – v SimuLTE o typy alokací a v ns-3 modulu LTE o algoritmy pro opakované použití frekvencí.

V poslední praktické části diplomové práce jsem srovnal frameworky z hlediska výkonnosti, efektivity a rozsahu funkcí, které nástroje pro potřeby vyhodnocení LTE sítí poskytují. Navrhnul jsem rozsáhlý a dynamický simulační scénář s možností nastavení libovolného počtu eNodeB. Dále jsem pomocí tohoto scénáře měřil závislosti výpočetní doby, využití operační paměti a procesoru na proměnných parametrech simulace. Jako proměnné parametry jsem zvolil počet eNodeB, velikost generovaného provozu v simulaci za sekundu a simulační dobu. Frameworky jsem následně srovnal z pohledu efektivity práce a přehledu možných funkcí, které dané nástroje poskytují uživateli. Každý nástroj nabízí své specifické možnosti pro potřeby vyhodnocení LTE sítí, které nemusí být součástí druhého nástroje.

Použitá literatura

[1] VIRDIS, Antonio; STEA, Giovanni; NARDINI, Giovanni. SimuLTE-A modular system-level simulator for LTE/LTE-A networks based on OMNeT++. In: [i]2014 International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH).[/i] IEEE, 2014. p. 59-70.

[2] *Github: inet-framework / SimuLTE* [online]. 20.8.2020 [cit. 2020-12-20]. Dostupné z: https://github.com/inet-framework/SimuLTE.

[3] *SimuLTE: LTE User Plane Simulation Model for INET & OMNeT++* [online]. [cit. 2020-12-20]. Dostupné z: https://SimuLTE.com/index.html

[4] *OMNeT++: Simulation manual* [online]. [cit. 2020-12-20]. Dostupné z: https://doc.omnetpp.org/omnetpp/manual/#sec:introduction:what-is-omnetpp

[5] *INET framework: what is inet framework* [online]. [cit. 2020-12-20]. Dostupné z: https://inet.omnetpp.org/Introduction.html

[6] NARDINI, Giovanni. Modeling Network-Controlled Device-to-Device Communications in SimuLTE. *Mdpi sensors* [online]. [cit. 2021-01-26]. Dostupné z: https://www.mdpi.com/1424-8220/18/10/3551/htm

[7] NARDINI, Giovanni, Antonio VIRDIS a Giovanni STEA. *Simulating device-to-device communications in OMNeT++ with SimuLTE: scenarios and configurations* [online]. Pisa, Italy [cit. 2021-01-29]. Dostupné z: https://arxiv.org/ftp/arxiv/papers/1609/1609.05173.pdf

[8] NARDINI, Giovanni, Antonio VIRDIS a Giovanni STEA. *Modeling X2 backhauling for LTE-Advanced and assessing its effect on CoMP Coordinated Scheduling* [online]. Pisa, Italy [cit. 2021-01-29]. Dostupné z: http://www.iet.unipi.it/a.virdis/publications/X2_modeling-openAccess.pdf

[9] *OMNeT++ Technical Articles* [online]. [cit. 2021-02-08]. Dostupné z: https://docs.omnetpp.org/tutorials/tictoc/part1/

[10] OMNeT++ User Guide. Omnetpp [online]. [cit. 2021-02-08]. Dostupné z: https://doc.omnetpp.org/omnetpp/UserGuide.pdf

[11] *Ns-3 manual: ns-3 project* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/release/3.32/manual/ns-3-manual.pdf

[12] *Ns-3: about* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/about/

[13] *Ns-3: LTE module documentation* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/html/lte-design.html

[14] *NS-3: ns-3 model library* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/ns-3-model-library.pdf#figure.19.1

[15] *NS-3: ns-3 model library* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/ns-3-model-library.pdf

[16] *NS-3: ns-3 model library* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/ns-3-model-library.pdf#figure.19.8

[17] *An LTE module for the ns-3 network simulator* [online]. Leden 2011 [cit. 2021-02-15]. Dostupné z: https://www.researchgate.net/publication/267227975_An_LTE_module_for_the_ns-3_network_simulator

[18] *NS-3: ns-3 model library* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/ns-3-model-library.pdf#capitole.19.1.9
[19] *Ns-3 manual: ns-3 project: fractional frequency reuse* [online]. [cit. 2021-02-15]. Dostupné z: https://www.nsnam.org/docs/models/html/lte-design.html#fractional-frequency-reuse
[20] Ns-3 Animation: model library [online]. [cit. 2021-03-02]. Dostupné z: https://www.nsnam.org/docs/models/html/animation.html
[21] Nsnam wiki: PyViz [online]. [cit. 2021-03-02]. Dostupné z: https://www.nsnam.org/wiki/PyViz
[22] Nsnam wiki: NetAnim 3.108 [online]. [cit. 2021-03-02]. Dostupné z: https://www.nsnam.org/wiki/NetAnim_3.108
[23] Dubai Burj Khalifas: Simulation Of 4g Lte Network With Ns 3 Simulator [online]. [cit. 2021-03-02]. Dostupné z: https://www.nsnam.org/wiki/NetAnim_3.108
[24] *OMNeT++ installation Guide* [online]. 2016 [cit. 2021-11-13]. Dostupné z: https://doc.omnetpp.org/omnetpp/InstallGuide.pdf
[25] *SimuLTE installation guide* [online]. 2015 [cit. 2021-11-13]. Dostupné z: https://SimuLTE.com/install.html
[26] *INET framework installation guide* [online]. 2021 [cit. 2021-11-13]. Dostupné z: https://inet.omnetpp.org/Installation.html
[27] *NS-3 random variables* [online]. 2021 [cit. 2021-11-15]. Dostupné z: https://www.nsnam.org/docs/manual/html/random-variables.html
[28] *Ns3 installation* [online]. 2021 [cit. 2021-11-17]. Dostupné z: https://www.nsnam.org/wiki/Installation#Installation
[29] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Physical layer procedures (3GPP TS 36.213 version 12.3.0 Release 12)* [online]. 2014 [cit. 2022-01-15]. Strana: 89 Dostupné z: https://www.etsi.org/deliver/etsi_ts/136200_136299/136213/12.03.00_60/ts_136213v120300p.pdf
[30] *REPORT ITU-R M.2135-1 - Guidelines for evaluation of radio interface technologies for IMT-Advanced - TABLE 8-2, str. 14* [online]. 2010, 72 [cit. 2022-01-29]. Dostupné z: https://www.itu.int/dms_pub/itu-r/opb/rep/R-REP-M.2135-1-2009-PDF-E.pdf
[31] *3rd Generation Partnership Project; Technical Specification Group Services and System Aspects; Policy and charging control architecture* [online]. Internet, 2016 [cit. 2022-03-03]. Dostupné z: https://www.arib.or.jp/english/html/overview/doc/STD-T63V12_10/5_Appendix/Rel13/23/23203-d80.pdf
[32] *LTE; Evolved Universal Terrestrial Radio Access (E-UTRA); Requirements for support of radio resource management (3GPP TS 36.133 version 13.3.0 Release 13)* [online]. 2016 [cit. 2022-03-20].Dostupné z: https://www.etsi.org/deliver/etsi_ts/136100_136199/136133/13.03.00_60/ts_136133v130300p.pdf

Seznam příloh

Struktura elektronické přílohy

Příloha A: *Architektura a simulační scénář pro handover v SimuLTE*
- *demo.xml*
- *handoverNetwork.ned*
- *omnetpp.ini*
- *run*

Příloha B: *Architektura a simulační scénář pro interferenci v SimuLTE*
- *demo.xml*
- *interferenceNetwork.ned*
- *omnetpp.ini*
- *run*

Příloha C: *Architektura a simulační scénář pro měření výkonnosti v SimuLTE*
- *demo.xml*
- *omnetpp.ini*
- *performanceNetwork.ned*
- *run*

Příloha D: *Seznam všech měřitelných statistik v SimuLTE*
- *meritelne_statistiky_SimuLTE.odt*

Příloha E: *Architektura a simulační scénář pro měření výkonnosti v ns-3*
- *performanceSimulation_NS3.cc*

Příloha F: *Architektura a simulační scénář pro handover v ns-3*
- *simulationHandover_NS3.cc*

Příloha G: *Architektura a simulační scénář pro interferenci v ns-3*
- *simulationInterference_NS3.cc*

Příloha H: *Python skript pro monitorování běhu procesu, respektive programu*
- *stats_collector.py*

Příloha I: *Seznam všech měřitelných zdrojů v ns-3 modulu LTE*
- *trasovaci_zdroje_ns-3_LTE.odt*