VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ
ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION
DEPARTMENT OF TELECOMMUNICATIONS

MODELOVÁNÍ BEZDRÁTOVÉHO PŘENOSOVÉHO KANÁLU V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER

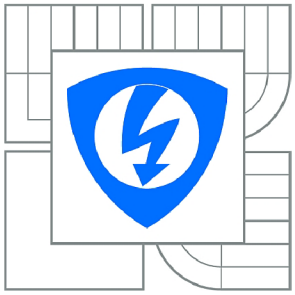
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE — Bc. SVĚTLANA SPIGLAZOVA
AUTHOR

BRNO 2012

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

MODELOVÁNÍ BEZDRÁTOVÉHO PŘENOSOVÉHO KANÁLU V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER

MODELLING THE WIRELESS TRANSMISSION CHANNEL IN OPNET MODELER SIMULATION ENVIRONMENT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

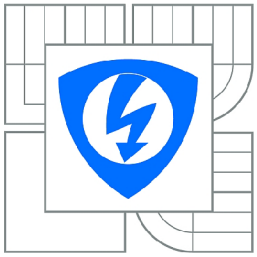
AUTOR PRÁCE — Bc. SVĚTLANA SPIGLAZOVA

AUTHOR

VEDOUCÍ PRÁCE — Ing. JIŘÍ HOŠEK, Ph.D.

SUPERVISOR

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Studentka: Bc. Světlana Spiglazova ***ID:*** 131612
Ročník: 2 ***Akademický rok:*** 2011/2012

NÁZEV TÉMATU:

Modelování bezdrátového přenosového kanálu v simulačním prostředí OPNET Modeler

POKYNY PRO VYPRACOVÁNÍ:

V rámci řešení diplomové práce důkladně prostudujte možnosti modelování bezdrátových technologií v simulačním prostředí OPNET Modeler. Dále se zaměřte na metody šíření signálu v bezdrátovém prostředí a způsoby ovlivnění parametrů tohoto přenosu. Podrobně popište obecný model vysílače bezdrátové technologie v OPNET Modeler a způsob přenosu datových jednotek bezdrátovým prostředím. Následně zdokumentujte také model přijímače bezdrátové technologie a způsob výpočtu základních parametrů přenosu signálu v bezdrátovém prostředí (SNR, chybovost, zpoždění, atd.). V prostředí OPNET Modeler vytvořte základní model bezdrátové sítě a nakonfigurujte v něm výměnu datových jednotek. Popište možnosti úpravy parametrů bezdrátového přenosového kanálu a následně proveďte sérii simulací s různě nastavenými parametry přenosového kanálu. V další části práce definujte speciální typ datové jednotky a pomocí ní přeneste mezi komunikujícími uzly hodnotu vybraného parametru rádiového signálu. Proveďte analýzu získaných výsledků a vše důkladně zdokumentujte v závěrečné zprávě.

DOPORUČENÁ LITERATURA:

[1] GAST, M.: 802.11 Wireless Networks: The Definitive Guide, Second Edition. Sebastopol: O'Reilly Media, 2005, ISBN: 978-0596100520.
[2] OPNET Technologies, OPNET Modeler Product Documentation Release 16.0, 2010.
[3] SINCLAIR, J.: How Radio Signals Work. New York: McGraw-Hill, 1998, ISBN: 978-0070580589.

Termín zadání: 6.2.2012 ***Termín odevzdání:*** 24.5.2012

Vedoucí práce: Ing. Jiří Hošek, Ph.D.

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Autor diplomové práce nesmí při vytváření diplomové práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ANOTACE

Tato diplomová práce se zabývá modelováním bezdrátového přenosu v simulačním prostředí OPNET Modeler.

V teoretické části je popsána samotná podstata šíření signálu prostřednictvím elektromagnetického vlnění včetně vlivů prostředí, které na šíření působí. Tyto vlivy ovlivňují popsané parametry přenášeného signálu, které charakterizují jeho kvalitu.

Dále je v teoretické části popsáno prostředí OPNET Modeler umožňující bezdrátový přenos navrhnout a prostřednictvím simulace ověřit jeho funkčnost a získat jeho parametry.

Samotný přenos signálu od vysílače k přijímači je rozdělen na samostatné kroky, které jsou detailně popsány s ohledem na implementaci ve výše uvedeném prostředí.

V praktické časti je nakonfigurován základní model bezdrátové sítě a provedena série simulací s různými variantami běžně nastavitelných základních parametrů přenosu jako jsou modulace signálu, typ antény a topologie sítě.

Dále je pozornost zaměřena na samostatné kroky přenosu, filosofii jejich spolupráce založenou na sdílení informací prostřednictvím datových jednotek zvaných TDA a jejich realizaci ve zdrojovém kódu jazyka C++ formou tzv. modelů přenosu.

Na základě těchto poznatků je nad rámec předem definovaných TDA vytvořen nový uživatelský TDA a prostřednictvím simulace je ověřena jeho správná implementace v rámci přenosu paketů.

Hodnoty vybraných TDA včetně uživatelských TDA jsou pomocí příkazů jazyka C++ vypsány do okna simulační konzole.

Získané výsledky simulací jsou na závěr analyzovány a zdokumentovány. Zdrojové kódy jsou k dispozici v přílohách diplomové práce.

Klíčová slova: OPNET Modeler, modelování, bezdrátová síť, bezdrátový přenos, TDA, C++

ABSTRACT

This diploma thesis deals with modeling wireless transmission in simulation environment OPNET Modeler.

In theoretical part, there is described the very substance of spreading the signal via electromagnetic wave including impact of the surroundings that affects the spreading. These impacts have an influence on described selected parameters of wireless transmission of the signal characterizing its quality.

Furthermore, there is described the OPNET Modeler setting, which allows designing the wireless transmission and by means of simulation verifies its functionality and acquires it is parameters.

The transmission of the signal from transmitter to receiver itself is divided to individual steps, which are described in detail considering the implementation in aforementioned environment.

In practical part, there is configured the basic model of wireless network and also executed series of simulations with different variants of commonly adjustable basic parameters of the transmission, as signal modulation, antenna types and network topology.

Additionally, attention is turned to the stages of transmission itself, philosophy of their cooperation based on sharing the information via data units called TDA and their implementation in source code of the C++ language called transmission models.

Based on these foundations is beyond beforehand defined created new user TDA and by means of simulation verified its correct implementation within packet switching.

Values of the selected including user TDA are written in the window of simulation console with the help of the C++ language.

Acquired results of the simulations are analyzed and documented in the conclusion. Source codes are available in the diploma thesis appendix.

Keywords: OPNET Modeler, modeling, wireless network, TDA, C++

SPIGLAZOVA, S. *Modelování bezdrátového přenosového kanálu v simulačním prostředí OPNET Modeler*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2012. 109 s. Vedoucí diplomové práce Ing. Jiří Hošek, Ph.D..

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Modelování bezdrátového přenosového kanálu v simulačním prostředí OPNET Modeler“ jsem vypracovala samostatně pod vedením vedoucího diplomové práce s použitím odborné literatury a dalších informačních zdrojů, které jsou uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušila autorská práva třetích osob, zejména jsem nezasáhla nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědoma následků porušení ustanovení §11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení §52 trestního zákona č. 140/1961 Sb.

V Brně dne..... .....

podpis autora

PODĚKOVÁNÍ

Děkuji vedoucímu práce Ing. Jiřímu Hoškovi, PhD. za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.

V Brně dne..... .....

podpis autora

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Purkynova 118, CZ-61200 Brno, Czechia

http://www.six.feec.vutbr.cz

Výzkum popsaný v této diplomové práci byl realizován v laboratořích podpořených z projektu SIX; registrační číslo CZ.1.05/2.1.00/03.0072, operační program Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007-13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKU

SEZNAM TABULEK

ÚVOD

Ačkoli se historie bezdrátové komunikace začala psát již před více než 100 lety, v současné době s nástupem výpočetní techniky nachází stále nová uplatnění. Jen těžko si lze dnes představit život bez mobilních telefonů, satelitní televize nebo bezdrátového připojení počítačů do sítě. Zvláště posledně jmenované prochází neustálým vývojem směřujícím k vyšší rychlosti a spolehlivosti a tím spojenému rostoucímu užívání těchto technologií.

Vzhledem k tomu, že pro bezproblémový provoz bezdrátové komunikace je nutno uvažovat mnoho rozličných faktorů, není divu, že pro rychlejší a jednodušší nasazení bezdrátových systémů do praxe byly podobně jako pro jiné obory navrženy specializované SW nástroje umožňující projektantům simulaci návrhu a následnou optimalizaci bezdrátové sítě.

Jedním z takových je OPNET Modeler (dále jen OM) firmy OPNET Technologies Inc. Cílem této diplomové práce je podrobný popis možností modelování bezdrátových technologií právě v tomto prostředí doplněný analýzou metod šíření signálu v bezdrátovém prostředí a způsobů ovlivnění parametrů tohoto přenosu.

V teoretické části je stručně vysvětlen princip bezdrátové komunikace a následně představen OM se všemi svými možnostmi. Simulace přenosu je zde rozdělena na samostatné kroky reprezentující vysílací a přijímací uzel. V těchto krocích probíhají výpočty typické pro každý krok jako např. poměr signál-šum SNR nebo bitová chybovost BER. Realizovány jsou prostřednictvím zdrojových kódů v programovacím jazyce C++ nazývaných modely, které jsou k práci přiloženy. Komunikace mezi jednotlivými modely je zajištěna pomocí speciálních datových jednotek zvaných TDA (Transmission Data Atribute). TDA je proměnná vybraného datového typu, která je zpravidla vázána na jednotlivé datové pakety přenosu. V prostředí OM je předdefinována skupina základních TDA zajišťující přenos základních parametrů. Kromě výše zmíněných umožňuje OM vytvořit nové uživatelské TDA dle potřeb projektu. Všechny TDA jsou v práci podrobně popsány včetně možností jejich čtení a nastavení.

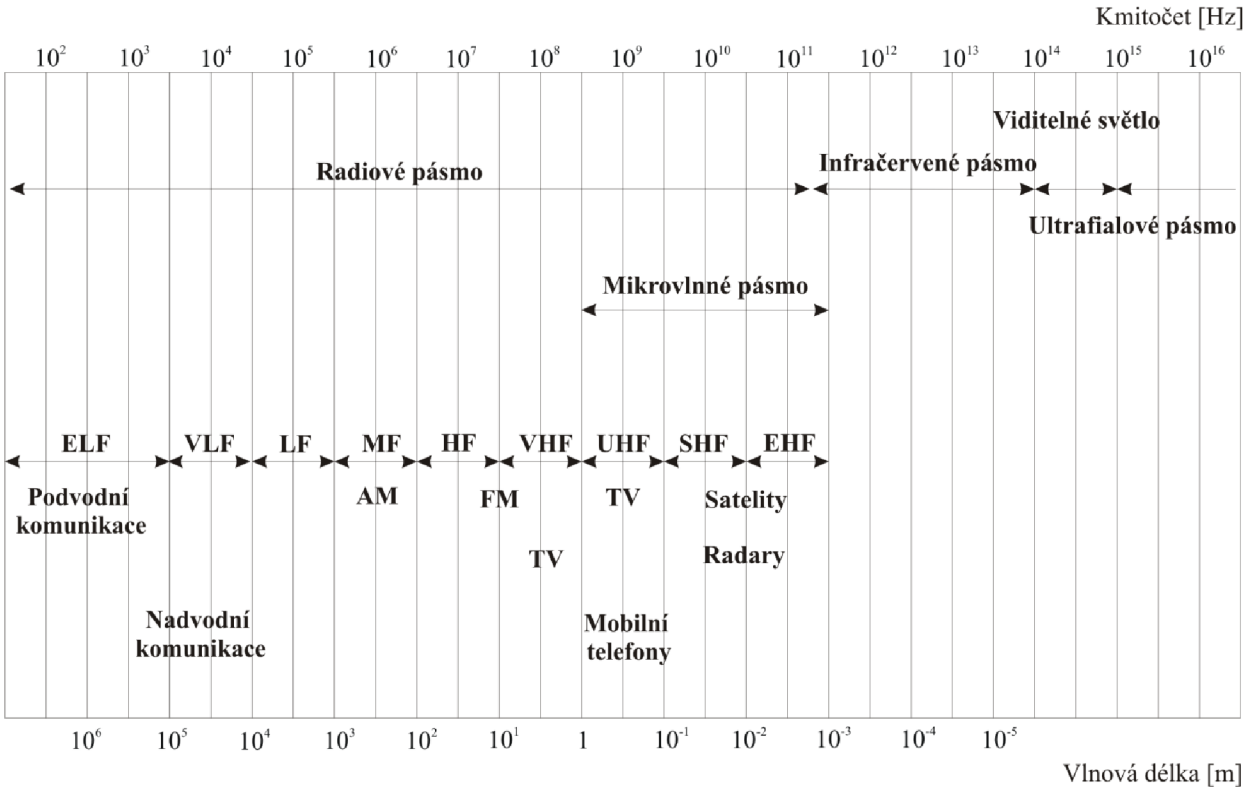
V praktické časti je nakonfigurován základní model bezdrátové sítě a provedena série simulací s různými variantami běžně nastavitelných základních parametrů přenosu jako jsou modulace signálu, typ antény a topologie sítě. Dále je pozornost zaměřena na práci s výše zmiňovanými datovými jednotkami TDA v rámci modelů přenosu. Kromě výchozích TDA byly vytvořeny nové TDA a ověřena jejich správná funkce sdílení informací mezi kroky přenosu. Vybrané TDA jsou v průběhu simulace prostřednictvím příkazů vypisovány do okna simulační konzole. Všechny získané výsledky simulací jsou v závěru náležitě dokumentovány a diskutovány.

1. BEZDRÁTOVÉ SYSTÉMY A METODY ŠÍŘENÍ SIGNÁLU

Komunikace bezdrátových zařízení probíhá mezi vysílačem a přijímačem, přičemž přenosová rychlost se pohybuje řádově v Mbps, avšak s neustálým vývojem lze očekávat rychlosti až do několika Gbps. Mezi bezdrátové systémy patří všechny mobilní sítě GSM (Global System for Mobile Communications), CDMA (Code Division Multiple Access), UMTS (Universal Mobile Telecommunication System), WiMAX (Worldwide Interoperability for Microwave Access) IEEE (Institute of Electrical and Electronics Engineers) 802.16, LTE (3GPP Long Term Evolution), atd..), osobní sítě (Bluetooth, ZigBee, a atd.), satelitní sítě a bezdrátové sítě podle standardu IEEE 802.11 známé zejména pod označením WLAN (Wireless Local Area Network), které se stále častěji používají pro připojení stanic do sítě LAN (Local Area Network) nebo Internetu.

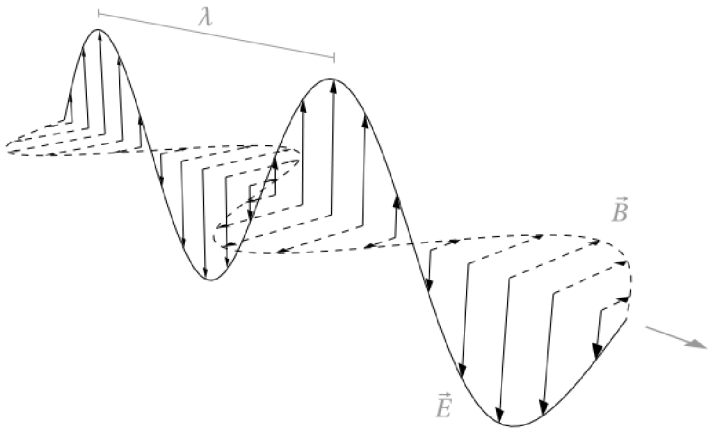
1.1 Elektromagnetické vlnění

K bezdrátovému přenosu signálu se využívá elektromagnetické vlnění. Lze jej využít v rozsahu infračerveného záření nebo například v podobě laseru. Nejvýznamnější použití je však v rozsahu radiového pásma, na které je tato práce zaměřena. Pásmo je definováno v rozsahu 10^3 až 10^8, jak je ukázáno na obr. 1.1.



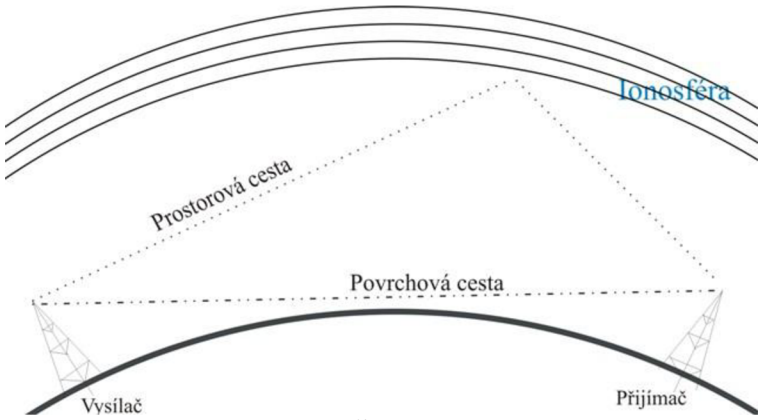
Obr. 1.1 Rozsah elektromagnetického spektra

Elekromagnetické vlnění lze chápat jako vlny definované rychlostí šíření, vlnovou délkou a frekvencí. Užitečný signál ve vlně je přenášen pomocí elektromagnetického pole popsaného intenzitami dvou na sebe kolmých složek – elektrickou a magnetickou, jak ukazuje obr. 1.2. Tento signál se šíří všemi směry rychlostí světla [10].



Obr. 1.2 Elektromagnetická vlna

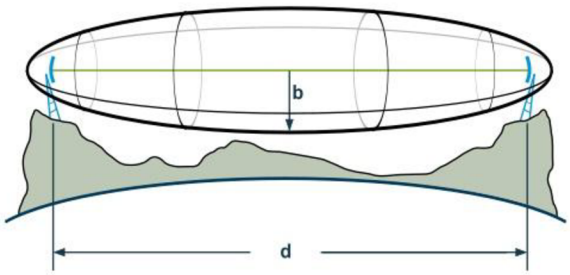
Signál lze zprostředkovávat dvěmi cestami, a to prostorovou a povrchovou, jak ukazuje obr. 1.3.



Obr. 1.3 Šíření signálu

Prostorový kanál je využíván pro vysoké kmitočty, jelikož s rostoucím kmitočtem klesá schopnost vlnění ohýbat se kolem zemského povrchu, čehož se využívá u šíření vlnění nižších frekvencí.

U síťových technologií v pásmu 2,4GHz je při přenosu vysokých frekvencí na krátké vzdálenosti nutné dodržet podmínku přímé viditelnosti vysílací a přijímací antény. Jak je naznačeno na obr. 1.4, pro bezproblémový provoz to není podmínka jediná. Spojnici obou antén obepíná tzv. Fresnelova zóna, jež by měla být nenarušena minimálně ze 60-ti procent.



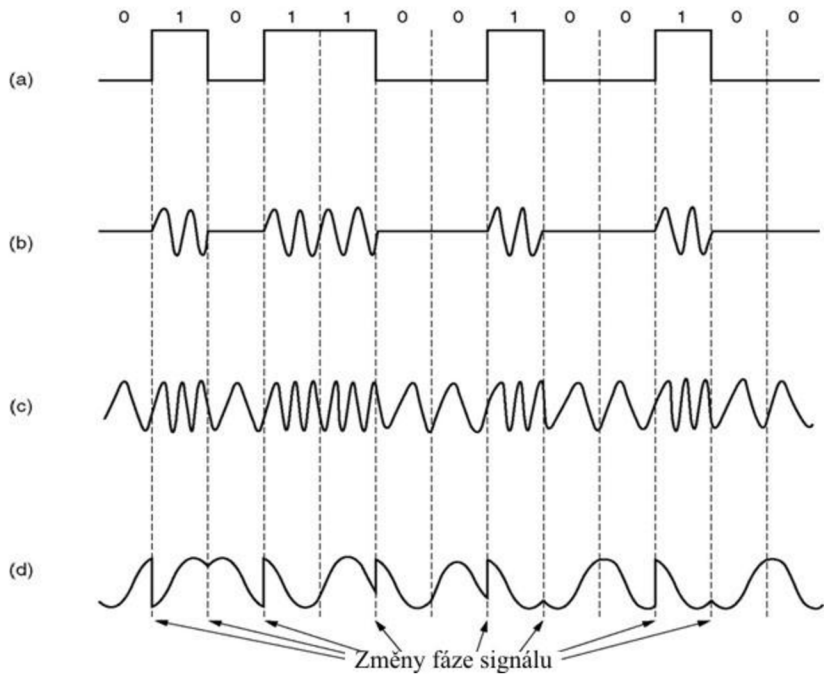
Obr. 1.4 Fresnelova zóna

1.2 Modulace signálu

Aby bylo dosaženo co možná nejefektivnějšího využití přenosového kanálu, je výhodné použít modulaci, kde je nosná vlna o vyšší frekvenci modulována úrovněmi signálu některé z modulací v základním pásmu. K modulaci je používán analogový nebo číslicový signál, kterým je modulována amplituda, kmitočet nebo fáze nosné vlny. Pro potřeby této práce budou vysvětleny digitální metody modulace. Modulovaný parametr této vlny se mění mezi dvěmi diskrétními stavy, z nichž jeden odpovídá modulačnímu bitu 0 a druhý bitu 1, z toho důvodu se tento způsob modulace označuje pojmem klíčování nebo zkratkou SK (Shift Keying). Základní digitální modulační techniky jsou:

- ASK (Amplitude-Shift Keying),
- FSK (Frequency-Shift Keying)
- PSK (Phase-Shift Keying) atd.

Princip modulací je zřejmý z obr. 1.5, kde signál (a) je modulační signál, (b) je modulovaný signál ASK, (c) je modulovaný signál FSK a (d) je modulovaný signál PSK. [5, 8]



Obr. 1.5 Typy modulace

1.3 Vliv prostředí na kvalitu přenosu

Při průchodu elektromagnetického vlnění volným prostorem se projevují vlivy, které způsobují komplikace příjmu relativně malých užitečných signálů nebo zvyšují bitovou chybovost a tím způsobují pokles přenosové rychlosti. Mezi nejčastější příčiny rušení a útlumu rádiových vln patří:

- **Odraz (reflexe)** – odraz vln může nastat na rozhraní dvou dielektrických prostředí s různými permitivitami. Platí zde Snellův zákon popisující šíření vlnění. Odraz radiových vln způsobuje i dokonale vodivé prostředí, např. parabolická anténa, jejíž plocha je mnohem větší než vlnová délka. [2, 6]
- **Lom (refrakce)** – stejně jako v předchozím případě, elektromagnetická vlna se láme vlivem přechodu z jednoho prostředí do druhého v závislosti na indexu lomu. Příkladem je lom vlny nad zemským povrchem vlivem změny hustoty a vlhkosti nad vodními plochami nebo lom ve vrstvách atmosféry. [2, 6]
- **Ohyb (difrakce)** – objeví-li se v trase šíření překážka, vlny se ohýbají i do prostoru za překážkou. Tento jev nastává tehdy, je-li vlnová délka záření mnohem větší než velikost překážky. [2, 6]
- **Rozptyl** – objevuje se při průchodu elektromagnetické vlny přes drobné překážky s rozměry mnohem menšími než její vlnová délka.
- **Mnohocestné šíření** – k přijímači přichází kromě vlny přímé i řada vln odražených, u kterých dochází ke zpoždění a ke změnám fáze a výkonové úrovně.
- **Ztráty šířením** – představují útlum, který závisí na délce spoje a typu prostředí. Do této skupiny patří i ztráty způsobené šířením volným prostorem.
- **Vliv atmosférických jevů** – do této skupiny se řadí déšť, mlha, sníh a jiné, při kterých dochází k útlumu elektromagnetické vlny.
- **Pomalé úniky** – útlum způsobený zastíněním spoje při pohybu antény. Ke kolísání přijímací úrovně dochází vzhledem k vlnové délce pomalu.
- **Rychlé úniky** – jsou způsobeny rychlým kolísáním úrovně signálu. Jsou způsobeny vícecestným šířením signálu a Dopplerovým posuvem, který vzniká v důsledku pohybu mobilní antény vůči okolním objektům. [13]

1.3.1 Útlum prostředí

Elektromagnetické vlny procházející přenosovým médiem jsou tlumeny v důsledku rozptylu vln do prostoru. Vztah pro výpočet útlumu volného prostoru FSPL (Free Space Path Loss) je uveden níže (1.1). Jednotkou jsou decibely.

$$FSPL = 20\log_{10}\frac{4\pi}{v} + 20\log_{10} f + 20\log_{10} d, \tag{1.1}$$

kde

f [Hz] je kmitočet elektromagnetické vlny,
d [m] je vzdálenost od vysílače,
v [m/s] je rychlost šíření světla (ve vakuu rychlost $\mathbf{c} = 2.99792458 \times 10^8$).

1.3.2 Poměr signál-šum (SNR)

Poměr signál-šum SNR (Signal to Noise Ratio) je obecně definován jako poměr výkonu užitečného signálu k výkonu šumu podle vzorce (1.2) a vyjadřuje odstup signálu od šumu.

$$SNR = \frac{P_{signal}}{P_{noise}}, \tag{1.2}$$

SNR bývá často vyjadřován v jednotkách dB podle vzorce (1.3).

$$SNR_{dB} = 10\log_{10}\left(\frac{P_{signal}}{P_{noise}}\right) = P_{signal,dB} - P_{noise,dB}, \tag{1.3}$$

1.3.3 Bitová chybovost (BER)

Bitová chybovost BER (Bit Error Rate) vyjadřuje četnost chybně přenesených bitů během přenosu a je definována poměrem chybně přijatých bitů k celkovému počtu přenesených bitů vztahem (1.4):

$$BER = \frac{bE}{vp \cdot t}, \tag{1.4}$$

kde

bE [bit] je počet chybně přijatých bitů
vp [bit/s] je přenosová rychlost
t [s] je doba měření

2 OPNET Modeler

Program OPNET Modeler (OM) je simulační prostředí vyvinuté firmou OPNET Technologies Inc. a slouží pro návrh, simulaci a analýzu různých síťových technologií a mechanismů takových jako modelování chování sítě, komunikačních protokolů, síťových prvků (servery, pracovní stanice, přepínače, směrovače, apod.), aplikací typu HTTP (Hypertext Transfer Protocol), FTP (File Transfer Protocol), email, VoIP (Voice over Internet Protocol), databáze, atd.

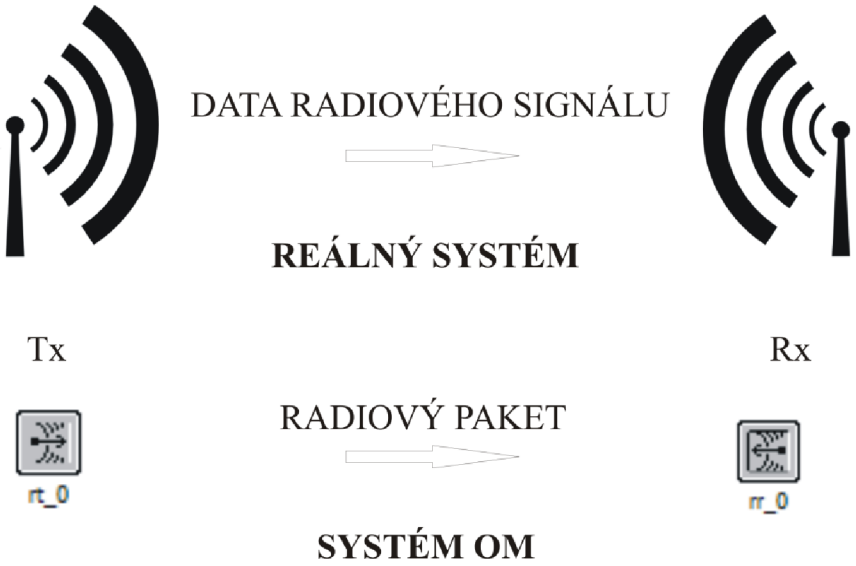
2.1 OPNET Modeler Wireless Suite

OM Wireless Suite poskytuje modelování, simulaci a analýzu bezdrátových sítí. Technologie umožňuje vývojářům využít pokročilé modelovací možnosti a bohatou sadu modelů protokolů pro návrh a optimalizaci svých bezdrátových protokolů, jako je stavební řízení a plánování algoritmů. Simulace odráží pohyb mobilních stanic v síti, včetně pozemních, vzdušných a družicových systémů. OM Wireless Suite podporuje všechny mobilní sítě (GSM, CDMA, UMTS, WiMAX IEEE 802.16, LTE, atd..) a také speciální bezdrátové sítě (IEEE 802.11), osobní sítě (Bluetooth, ZigBee, a atd.) a satelitní sítě.

OM umožňuje inženýrům analýzu chování kdekoli v síti, nastavení výkonu sítě, a vyhodnocení scénáře růstu síťových služeb, které přinášejí zisk.

3 POPIS KROKŮ RÁDIOVÉHO PŘENOSU

Tato část práce se zabývá procesem přenosu dat (paketů), který používá OM pro modelování bezdrátového prostředí. Pod označením paket zde bude rozuměn paket, rámec i signál definované pro reálné systémy (obr. 3.1). Použité texty jsou překladem z anglického manuálu k simulačnímu prostředí OM.



Obr. 3.1 Bezdrátový přenos dat v reálném systému a v OM

3.1 Úvod

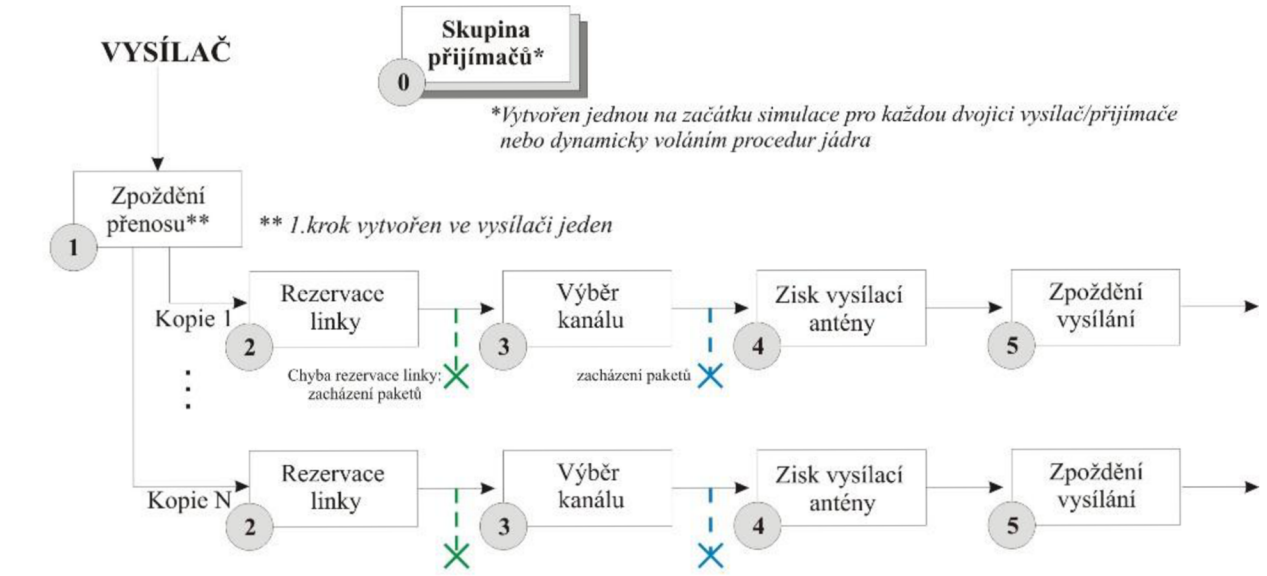
Při přenosu paketů z vysílače do přijímače se používá několik kanálů, přičemž každý přenos dat vyžaduje jiné chování a časování. V důsledku toho se předání paketů provádí pro každý přijímač zvlášť. Další podrobně popsané kroky přenosu paketů jsou na začátku ve výchozím nastavení vysílače a přijímače.

1. krok (zpoždění přenosu) se používá pro výpočet výsledku, který je společný pro všechny destinace, a proto může být proveden pouze jednou za přenos. Dále jsou vytvořeny kopie paketů v závislosti výsledku 2. kroku (rezervaci linky), který ověřuje ustanovení spojení mezí vysílačem a přijímačem. Podobně 3. krok (výběr kanálu) klasifikuje přenos s ohledem na konkrétní kanál přijímače a tím uděluje směr k dosažení závěrečného kroku.

Z obr. 3.2 a obr. 3.11 je zřejmé, že některé z pozdějších kroků radiového přenosu mohou být provedeny vícekrát pro každý přenos, díky interakci s více souběžnými přenosy z jiných zdrojů. Pro zjištění příjmu více paketů současně kanálem přijímače, simulace jádra udržuje dva seznamy „aktuálních“ paketů pro každý kanál. První seznam obsahuje pouze pakety, které byly stanoveny v platnosti do 3. kroku výběru kanálu, druhý seznam obsahuje neplatné pakety. Platné jsou ty pakety, které splňují kritéria pro správný přenos kanálu. Obecně platí, že je nutné, aby kanály přijímače a vysílače měly stejné vlastnosti, a případně také aby kanál přijímače byl zadán do paketů signálu. Hlavním důvodem pro zachování platnosti paketů v jednotlivých seznamech je, že umožňují jádru simulaci určitých kroků nebo výpočet statistiky kanálu pouze pro platné pakety. Například, není potřeba počítat poměr signál-šum, bitovou chybovost, nebo chybu alokace pro pakety, které nemohou být přijaty přijímačem. Obecně platí, že všechny kroky následující po 7. kroku se vztahují pouze pro platné pakety.

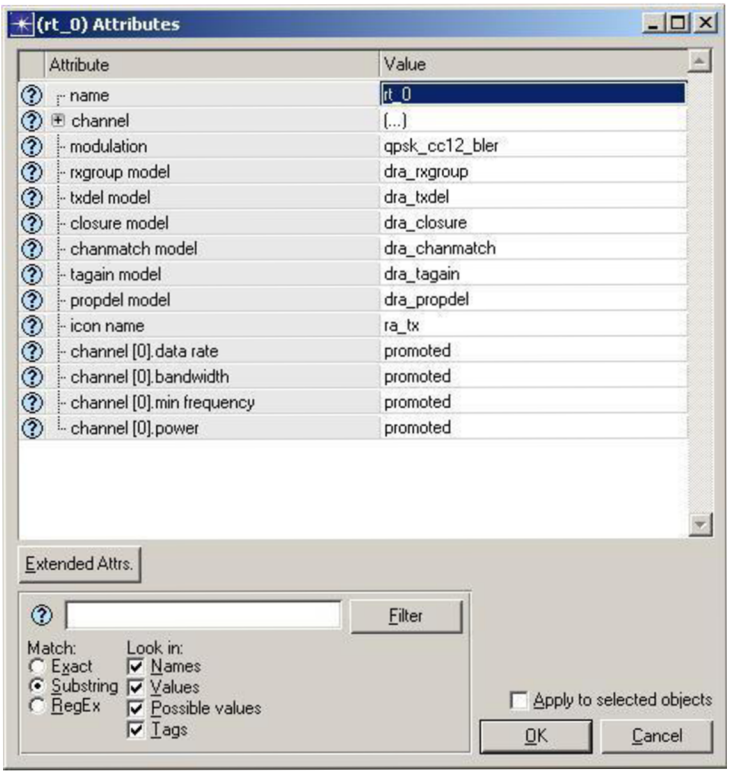
Kroky 9 až 12 se uplatňují při hodnocení vlastností propojení v reakci na změny ve stavu signálu. Je vždy alespoň jedno volání 10. až 12. kroku k posuzování vlastností po celou dobu platnosti paketu. Nicméně, další volání bude provedeno pro každý z těchto kroků (9 - 12), když přijde interferenční paket, dojde k výpočtu nových podmínek signálu.

Vzhledem k tomu, že rádiové spoje neexistují jako fyzické předměty, jednotlivé kroky sloužící k podpoře konkrétního rádiového přenosu musí být spojeny s vysílačem a přijímačem, který tvoří spojení. [7]



2.-5.kroky vytvořeny pro každý kanál ve skupině přijímače jednotlivě

Obr. 3.2 Kroky radiového přenosu vysílačů v prostředí OM



Obr. 3.3 Nastavení kroků přenosu vysílačů v prostředí OM

3.2 Popis kroků a výchozích modelů vysílače

0. KROK. SKUPINA PŘIJÍMAČŮ

Krok „Skupina přijímačů“ není součástí dynamického systému, který zpracovává přenosy. Nicméně, je považován za část přenosu, protože počítá výsledky, které ovlivňují chování vysílání. Je určený atributem vysílače *rxgroup model*.

Když rádiový přenos začíná, simulační jádro modeluje všesměrové vysílání s vloženými násobnými spoji mezi vysílačem a přijímačem. Každý vysílač vede vlastní skupinu přijímačů, které jsou možnými kandidáty pro příjem vysílání z tohoto objektu. Cílem tohoto kroku je vytvoření inicializační skupiny přijímačů pro každý kanál vysílače. Simulační jádro kontroluje všechny možné dvojice spojů vysílač/přijímač a vytvoří skupinu přijímačů pro každý kanál vysílače.

To ale představuje problém. Vzhledem k tomu, že charakteristiky vysílače se mohou dynamicky měnit v průběhu simulace, je často obtížné na začátku určit, které přijímače jsou vhodné cíle pro daný vysílač. Proto krok „skupiny přijímačů“ zahrnuje přijímače, pokud není určeno předem, že přijímač nikdy nebude vhodný cíl pro vysílač. Následné kroky mohou během simulace použít dynamická kritéria pro stanovení způsobilosti přijímače. Mezi možné důvody pro vyloučení přijímače během simulace patří:

- rozdělení frekvenčních pásem;
- fyzické oddělení;
- nulující anténa.

U některých modelů sítě může simulace jádra určit, že některé páry vysílač-přijímač jsou zcela neschopny komunikovat. V takových případech odebrání přijímače od vysílače vede k rychlejší simulaci, protože to vylučuje provedení spojů, u kterých je známo, že nemají smysl. Protože krok „Skupiny přijímačů“ se v simulaci nazývá krok 0, jeho výsledky nejsou závislé na faktorech, které se mohou měnit v průběhu simulace. Příkladem takovýchto potenciálně problémových kritérií jsou počáteční vzdálenost mezi dvěma mobilními uzly a počáteční frekvence přidělená vysílači a přijímači (za předpokladu, že model může dynamicky měnit tyto frekvence). V takových případech výchozí nastavení přijímače a všech skupin přijímačů zůstává statické po celou simulaci, dalších kroky pak používají dynamická kritéria pro hodnocení spojů vysílač-přijímač.

Ke změně výchozího chování kroku a dynamickým změnám a přepočtům skupin přijímačů v reakci na události simulace může být použit balík procedur jádra (Radio Package). Například může být odstraněn přijímač ze skupiny přijímačů v případě, že v průběhu simulace se přijímací uzel stane neaktivním. Může být použit postup *op_radio_txch_rxgroup_compute ()* pro výpočet nebo přepočet dané skupiny přijímačů kdykoli během simulace. Krok může být dokonce úplně „přeskočen“ a skupina přijímačů vytvořena a aktualizována podle potřeby.

Dynamická aktualizace skupin přijímačů může vést k rychlejším simulacím, zejména v síťových modelech s vysokou úrovní provozu. [7]

Volání modelu:

Tento model – model kroku „Skupina přijímačů“ se jmenuje *dra_rxgroup*. Zdrojový kód je k dispozici v souboru *dra_rxgroup.ps.c* (příloha b), který se nachází v adresáři */models/std/wireless*. Alternativní verze tohoto modelu pro sítě nepoužívající informace o stavu přijímaného kanálu se nachází v souboru *dra_rxgroup_no_rxstate.ps.c*. [7]

Model „Skupina přijímačů“ je v simulačním procesu volán pouze jednou v čase 0 z důvodu vyhodnocení spojení mezi všemi vysílacími a přijímacími uzly. Tím se liší od všech ostatních kroků přenosu, které jsou volány opakovaně pro každý paket. Nicméně je pomocí procedur jádra možné průběžně měnit skupiny přijímačů vysílacích kanálů. Tento krok vyžaduje dva argumenty. *Object ID* vysílacího a přijímacího kanálu.

Jelikož je tento krok volán v čase 0, je použit k inicializaci stavu každého přijímacího kanálu. Informace o stavu přijímacího kanálu se požívá k předávání informací mezi modely kroků radiového přenosu.

Výsledek kroku volaného pro vysílací a přijímací kanály je celé číslo vyjádřené dvěma možnými stavy *OPC_TRUE* nebo *OPC_FALSE*. Toto číslo vyjadřuje, zda je nebo není možné navázat komunikaci mezi vysílacím a přijímacím uzlem. [7]

Způsobilost přijímače

Účelem modelu je odstranit nezpůsobilé přijímací kanály s ohledem na jednotlivé vysílací kanály, čímž dojde ke zvýšení výkonu simulace. V defaultním modelu jsou za způsobilé považovány všechny přijímače. To znamená, že model vždy vrací hodnotu *OPC_TRUE*. [7]

1. KROK. ZPOŽDĚNÍ PŘENOSU

Tento krok „Zpožděni přěnosu“ je v pořadí druhým krokem přenosu, ale je první, který se vytváří dynamicky jako výsledek nového přenosu. Je určený atributem vysílače *txdel model.* To je jediný krok, jehož jedno vytvoření má význam na všech linkách nového přenosu, které vyplývají z nového vysílání.

Tento krok je volán k výpočtu celkového času potřebného pro dokončení přenosu celého paketu. Výsledek simulace je časový rozdíl mezi začátkem vyslání prvního bitu a koncem přenosu poslední části paketu. Simulační jádro používá výsledek poskytovaný tímto krokem pro plánování na konci přenosu pro vysílač, který se používá pro odeslání paketů. Když dojde k této události, vysílač může zahájit vysílání dalšího paketu ve vnitřní frontě, pokud je přítomen, jinak zůstává vysílač v nečinnosti. Navíc tento výsledek je použit ve spojení s výsledkem kroku zpoždění vysílání k výpočtu času, ve kterém paket dokončí příjem v místě určení. [7]

Volání modelu:

Tento model – model kroku „Zpožděni přěnosu“ se jmenuje *dra_txdel*. Zdrojový kód je k dispozici v souboru *dra_txdel.ps.c* (příloha c), který se nachází v *adresáři /models/std/wireless*.

Model „Zpoždění přenosu“ je v simulačním procesu volán simulačním jádrem ihned po začátku přenosu paketu. Jeho cílem je výpočet času pro vysílač potřebného pro kompletní zpracování a předání paketu. Jde o simulovaný časový interval oddělující začátek předání prvního bitu a konec přenosu posledního bitu paketu

Výsledek kroku volaného pro každý paket je 64 bitové číslo s plovoucí desetinnou čárkou, jež představuje zpoždění vysílání paketu. Simulační jádro očekává, že tato hodnota bude uložena pro každý paket v *TDA OPC_TDA_RA_TX_DELAY*. [7]

Výpočet zpoždění vysílání

Pro všechny přenášené pakety je zpoždění vysílání počítáno na základě rychlosti datového kanálu a délce paketu. Model nejdříve načte rychlost přenosového kanálu, která je k dispozici v TDA *OPC_TDA_RA_TX_DRATE*. Poté je získána délka paketu a vydělena rychlostí kanálu. Tato hodnota je uložena v TDA *OPC_TDA_RA_TX_DELAY*. [7]

2. KROK. REZERVACE LINKY

Krok „Rezervace linky" je třetí krok přenosu, a je specifikován atributem *closure model* radiového vysílání. Je volán pro každý přijímač s odkazem na nastavení cílového kanálu. Cílem tohoto kroku je určit, zda konkrétní přijímač může být ovlivněn přenosem. Schopnost přenosu dosáhnout přijímače je označována jako uzavření spojů mezi vysílačem a přijímačem.

Cílem kroku není určit, zda přenos je platný, nebo vhodný pro konkrétní kanál, ale pouze jestli přenášený signál může fyzicky dosáhnout potenciálního přijímače a ovlivnit ho jakýmkoliv způsobem, takže se tento krok týká rušivého vysílání, stejně jako požadovaného. Obecně platí, že výpočty v tomto kroku jsou prováděné především na základě fyzikálních úvah, jako vliv překážek anebo povrchu Země. [7]

Volání modelu:

Tento model – model kroku „Rezervace linky" se jmenuje dra_txdel. Zdrojový kód je k dispozici v souboru *dra_closure.ps.c* (příloha d), který se nachází v adresáři */models/std/wireless.*

Model je volán ihned po výpočtu zpoždění přenosu bez simulovaného zpoždění a určuje linku radiového přenosu mezi vysílačem a přijímačem. Simulovaný čas udává začátek přenosu paketu. Model rezervace linky umožňuje dynamicky zapínat nebo vypínat linky radiového přenosu. Jinými slovy všechny dvojice vysílacích a přijímacích kanálů, které jsou způsobilé, budou použity jednotlivě v samostatných voláních modelu. To umožňuje docílení např. omezeného rozsahu přenosu nebo dynamické změny atributů vysílačů a přijímačů.

Výsledek kroku volaného pro každý paket je logická hodnota *OPC_TRUE* nebo *OPC_FALSE*, jež představuje schopnost vysílače navázat spojení s přijímačem v době přenosu paketu. Simulační jádro očekává, že tato hodnota bude uložena pro každý paket v TDA *OPC_TDA_RA_CLOSURE*. Bude-li výsledkem modelu hodnota *OPC_FALSE*, tedy, že spojení nebude navázáno, simulační jádro nebude volat žádné další kroky radiového přenosu. [7]

Modelování terénu:

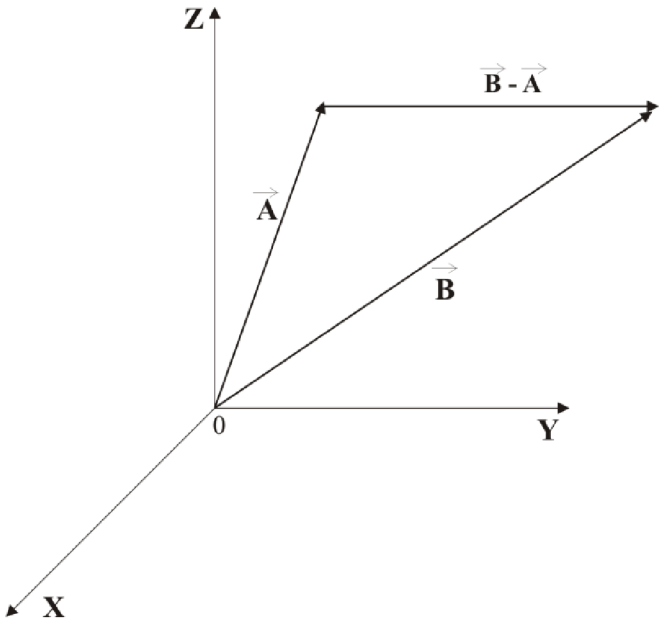
Výchozí model rezervace linky zahrnuje podporu modelování terénu pomocí OPNET Terrain Modeling Module (TMM), díky čemuž je rezervace linky založena na kritériích stanovených použitím modelu šíření. [7]

Výpočet rezervace linky:

Jestliže není použito modelování terénu TMM, je výpočet rezervace linky pro všechny pakety počítán na základě metody trasování paprsku *line-of-sight* algoritmu. Algoritmus testuje spojnici mezi vysílačem a přijímačem na zemském povrchu. Země je modelována jako koule.

Pokud spojnice prochází zemských povrchem, přijímače nemůže být dosaženo a další kroky přenosu nebudou volány. Tento algoritmus neuvažuje ohyb vln.

Line-of-sight algoritmus při výpočtu používá vysílací vektor, přijímací vektor a rozdílový vektor z vysílače do přijímače. Tyto vektory jsou definovány v souřadnicovém systému, jehož počátek leží ve středu Země. Vektory jsou popsány jako $\vec{v} = (x, y, z)$ a mohou být reprezentovány jako směrové úsečky začínající v libovolném bodě v prostoru (x', y', z') a končící v bodě $(x' + x, y' + y, z' + z)$. Vektorové sčítání může být provedeno jako sčítání dvojic vektorových souřadnic. Rozdíl mezi dvěma vektory $\vec{B} - \vec{A}$ je vektor, který při sečtení s vektorem $\vec{B}$ vytvoří vektor $\vec{A}$. Tyto vztahy jsou popsány níže pro dva vektory $\vec{A}$ a $\vec{B}$ začínající v počátku, viz obr. 3.4. [7]



Obr. 3.4 Reprezentace rozdílového vektoru

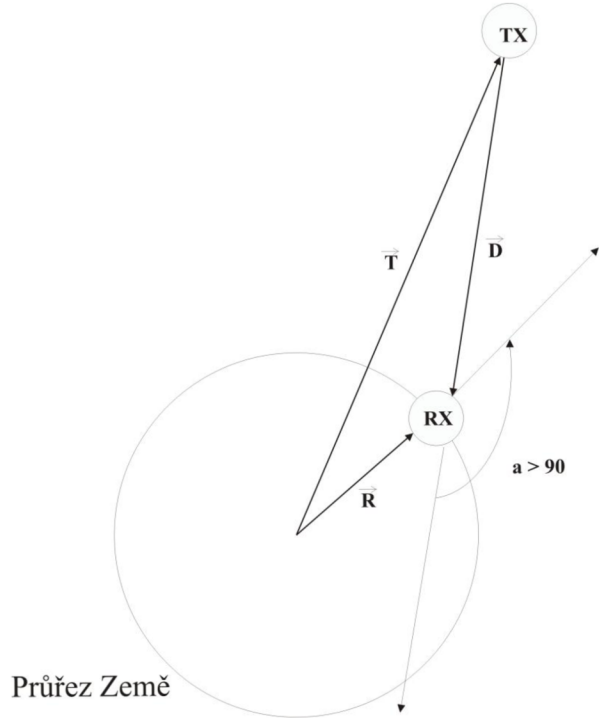
Vysílací vektor $\vec{T}$ je vektor začínající v počátečním bodě a končící v bodě umístění vysílače $(t_x,\ t_y,\ t_z)$. Obdobně přijímací vektor $\vec{R}$ může začínat v počátečním bodě a končící v bodě umístění přijímače $(r_x,\ r_y,\ r_z)$. Rozdílový vektor od vysílače k přijímači začíná v bodě $(t_x,\ t_y,\ t_z)$ a končí v bodě $(r_x,\ r_y,\ r_z)$. Jeho souřadnice tedy jsou $\vec{D} = (r_x - t_x, r_y - t_y, r_z - t_z)$. [7]

Funkce *dra_closure* používá *line-of-sight* algoritmus testující rozdílový vektor od vysílače k přijímači a jeho průnik povrchem Země. Model využívá skutečnosti, že k průniku nedojde v jednom ze tří následujících případů:

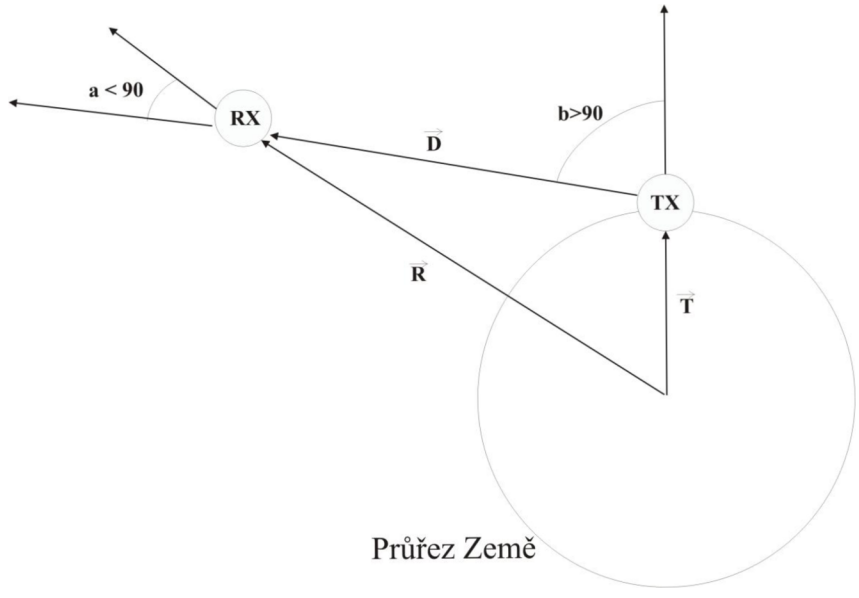
1. úhel mezi přijímacím vektorem $\vec{R}$ a rozdílovým vektorem $\vec{D}$ je větší než 90^o.
2. úhel mezi přijímacím vektorem $\vec{R}$ a rozdílovým vektorem $\vec{D}$ je menší nebo roven 90^o. A úhel mezi vysílacím vektorem $\vec{T}$ a rozdílovým vektorem $\vec{D}$ je menší než 90^o.
3. úhel mezi přijímacím vektorem$\vec{R}$a rozdílovým vektorem $\vec{D}$ je menší nebo roven 90^o., úhel mezi vysílacím vektorem $\vec{T}$ a rozdílovým vektorem $\vec{D}$ je větší nebo roven 90^o A délka

kolmice od spojnice vysílače a přijímače ke středu Země je větší než poloměr Země.

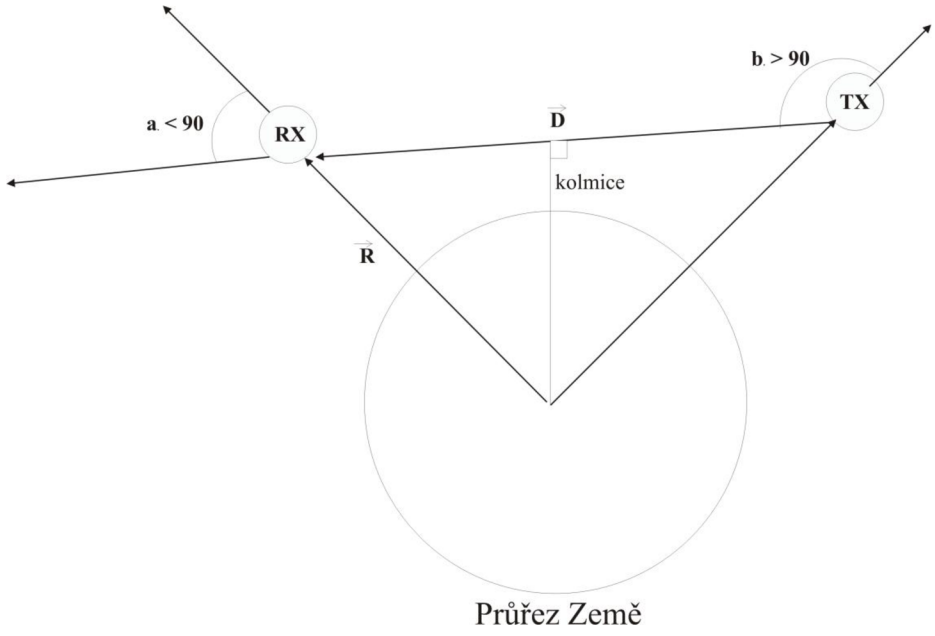
Příklad každého z těchto případů je znázorněn obr. 3.5, obr. 3.6 a obr. 3.7. Umístění vysílače a přijímače je označeno TX a RX. [7]



Obr. 3.5 Případ 1 – podmínky navázání spojení



Obr. 3.6 Případ 2 – podmínky navázání spojení



Obr. 3.7 Případ 3 – podmínky navázání spojení

Model rezervace linky počítá skalární součin vektorů $\vec{R}$ a $\vec{D}$ k určení velikosti úhlu mezi vektory. Skalární součin může být dán vztahem (3.1):

$$a \cdot b = \|a\|\|b\| \cos\theta. \tag{3.1}$$

proto tedy
$a \cdot b = 0$ když $\cos\theta = 0$ (tedy když $\theta = 90$ nebo 270)
$a \cdot b > 0$ když $\cos\theta > 0$ (tedy když $\theta < 90$ nebo $\theta > 270$)
$a \cdot b > 0$ když $\cos\theta > 0$ (tedy když $90 < \theta < 270$)

3. KROK. VÝBĚR KANÁLU

Krok „Výběr kanálu“ je čtvrtým krokem přenosu a je popsán atributem vysílače *chanmatch model*. Je volán pro každý kanál přijímače, který splňuje kritéria kroku „Rezervace linky. Cílem tohoto kroku je klasifikovat přenos s ohledem na přijímače. Jedna ze tří možných kategorií definovaných níže musí být paketu přiřazena:

- **platný**. Pakety v této kategorii jsou považovány za slučitelné s přijímačem a bude možné je přijmout a předat do jiných modulů v přijímacím uzlu za předpokladu, že nebudou ovlivněny velkým množstvím chyb. Klasifikace typu platný paket obvykle závisí alespoň na dohodě mezi vysílačem a přijímačem o hodnotách některých klíčových atributů.
- **šum**. Tato klasifikace se používá k identifikaci paketů, jejichž datový obsah nemůže být přijat, ale které mají dopad na přijímač generováním interference. Pakety jsou obecně klasifikované jako šum v důsledku nekompatibility mezi konfigurací vysílače a přijímače.

- **ignorovaný**. Je-li přenos bez vlivu na přijímač, pak by měl být identifikován pomocí této klasifikace. Simulační jádro pak přeruší přenos mezi vysílačem a přijímačem tohoto vysílání.

Volání modelu:

Tento model- model kroku „Výběr kanálu“ se jmenuje *dra_chanmatch*. Zdrojový kód je k dispozici v souboru *dra_chanmatch.ps.c* (příloha e), který se nachází v adresáři */models/std/wireless*.

Krok výběru kanálu zajišťuje kompatibilitu mezi vysílacím a přijímacím kanálem. Model je volán ihned po dokončení předchozího kroku bez simulovaného zpoždění za předpokladu, že možnost navázání spojení existuje. Čas volání je počáteční čas přenosu paketu. Model se používá ke klasifikaci přenosu na tři kategorie s ohledem na požadavky přijímače. Pakety mohou být považovány za platné, které je přijímač schopen zpracovat, rušení, které nemůže být dekódováno a zpracováno, ale ovlivňuje další kroky přenosu, nebo ignorovány a přenos tohoto paketu je ukončen.

Výsledek kroku volaného pro každý paket je celé číslo udávající kategorii, do které daný paket patří. Těmto kategoriím odpovídají symbolické konstanty *OPC_TDA_RA_MATCH_VALID*, *OPC_TDA_RA_MATCH_NOISE* a *OPC_TDA_RA_MATCH_IGNORE*. Simulační jádro očekává, že tento výsledek bude uložen pro každý paket v TDA *OPC_TDA_RA_MATCH_STATUS*. [7]

Hodnocení kompatibility paketu

Model vyhodnotí kanály vysílače a přijímače za kompatibilní v případě, že mají společné charakteristiky. Tyto parametry jsou analyzovány a patří mezi ně přenosová frekvence, šířka pásma, rychlost přenosu dat, rozprostírací kód a modulace. Ty jsou porovnávány s atributy kanálu minimální frekvence, šířka pásma, rychlost přenosu dat, atribut modulace vysílače a přijímače atd. Jestliže se tyto atributy shodují, paket je považován za platný.

Jestliže se pásmo vyhodnocovaného paketu nepřekrývá s pásmem přijímacího kanálu, paket je považován za neplatný, a který není schopen významným způsobem přijímač ovlivnit. Proto je takový paket označen za ignorovaný a žádné další krok přenosu se již neprovedou.

Jestliže se pásmo vyhodnocovaného paketu překrývá s pásmem přijímacího kanálu, ale žádný z dalších atributů kanálu neodpovídá, paket je považován za rušení. Pakety rušení jsou zpracovávány pouze tehdy, pokud mohou ovlivnit správný příjem platných paketů. Proto některé následující kroky přenosu nebudou pro tyto pakety volány. [7]

4. KROK. ZISK VYSÍLACÍ ANTÉNY

Krok „Zisk vysílací antény“ je pátý krok přenosu a je určen atributem vysílače *tagain model*. Provádí se samostatně pro každý určený přijímač, kromě těch, které nesplnily kritéria kroku „Rezervace linky“ a těch, u nichž čtvrtý krok „výběr kanálu“ skončil označením „ignorovaný“.

Cílem tohoto kroku je stanovení zisku vysílače spojeného s anténou vycházející ze směru vektoru vedoucího od vysílače k přijímači. Simulační jádro samo o sobě nepoužívá tento výsledek, ale je typickým prvkem při výpočtu 7. kroku.

Na základě měření nebo výpočtu zisku v rozmezí směrů může být vytvořen trojrozměrný anténní obrazec, který charakterizuje efekt antény s ohledem na různé směry přenosu. [7]

Obecně platí, že krok počítá nejprve směr vektoru oddělující vysílač a přijímač, potom využívá znalostí anténního obrazce k určení zisku antény za předpokladu přenosu. Přenosové kroky mohou určit polohování antény čtením hodnot těchto atributů přímo z objektu antény.

Volání modelu:

Tento model – model kroku „Zisk vysílací antény“ se jmenuje *dra_tagain*. Zdrojový kód je k dispozici v souboru *dra_tagain.ps.c* (příloha f), který se nachází v adresáři */models/std/wireless*.

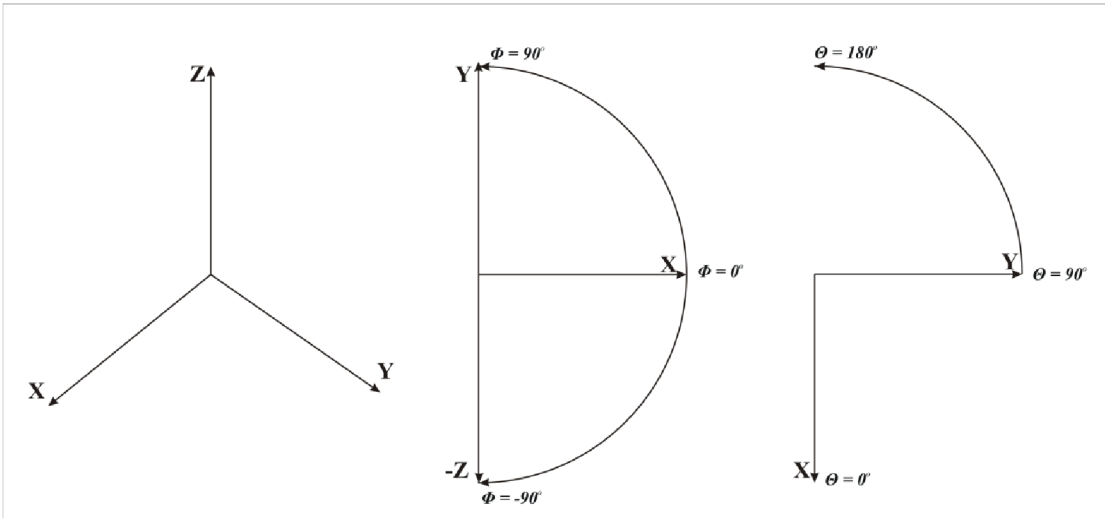
Za předpokladu, že paket nebyl ignorován, krok definuje zisk vysílací antény spojený s přenášeným paketem ihned po ukončení předchozího kroku bez simulovaného zpoždění. Čas volání je počáteční čas přenosu paketu.

Výsledek kroku volaného pro každý paket je hodnota s pohyblivou řádovou čárkou určující zisk vysílací antény ve směru k požadovanému přijímači. Tento výsledek je vyjádřen v dB a simulační jádro očekává, že tato hodnota bude uložena pro každý paket v TDA *OPC_TDA_RA_TX_GAIN*. [7]

Výpočet zisku vysílací antény

Pro všechny přenášené pakety počítá model zisk vysílací antény na základě vektoru mezi vysílačem a přijímačem a příslušných atributů spojených s anténou. Speciální případ je použití všesměrové antény, pro kterou žádné výpočty nejsou potřebné, protože zisk bude vždy 0dB.

Obecně model počítá vertikální a horizontální rovinné úhly *fi* (Φ) a *théta* (Θ) přijímače a s vysílačem v nulovém bodě souřadnicového systému. Jde o levotočivý souřadnicový systém s osami definovanými na obr. 3.8. Tento diagram zobrazuje také vertikální a horizontální rovinný úhel Φ a Θ, které model používá k určení směru. [7]



Obr. 3.8 Osy použitého souřadnicového systému

Model nejprve vypočítá rozdílový vektor $\vec{D}$ pomocí vysílacího a přijímacího vektoru. Tím je získán vektor od vysílače k přijímači. Atributy související s body antény jsou načteny. První dvojice rovinných úhlů *OPC_TDA_RA_TX_PHI_POINT* a *OPC_TDA_RA_TX_THETA_POINT* představují

směr vysílací antény z hlediska odchylek Φ a Θ. Tyto hodnoty jsou uvedeny ve stupních a jsou počítány simulačním jádrem na základě umístění přijímače prostřednictvím zeměpisné šířky, zeměpisné délky a nadmořské výšky. Další dvojice atributů *OPC_TDA_RA_TX_BORESIGHT_PHI* a *OPC_TDA_RA_TX_BORESIGHT_THETA* představuje směr paprsku vysílací antény. Druhá dvojice atributů je definována v samostatném souřadnicovém systému, který se liší od potřeb pro použití v modelech přenosu. Proto je nutný přepočet úhlu Φ.

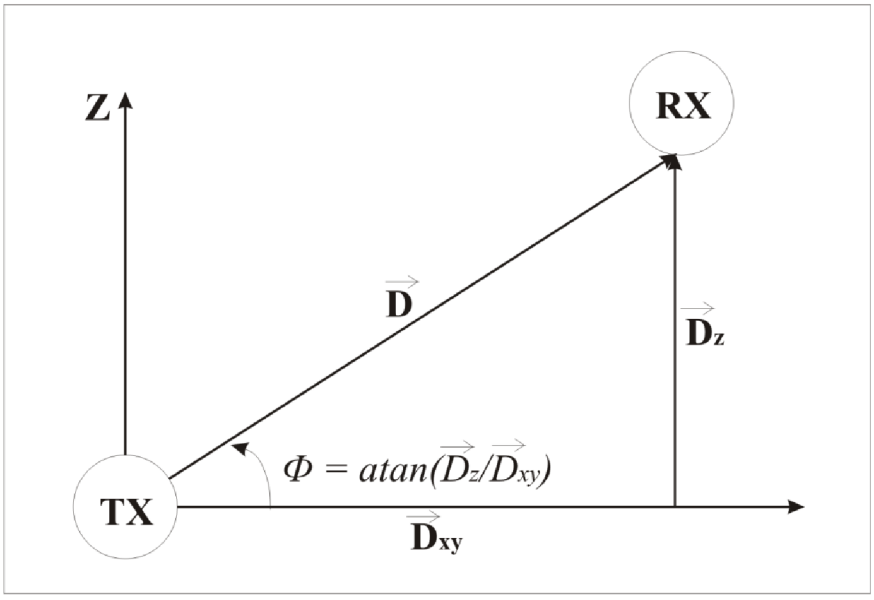
Rotace úhlu Φ lze nejsnáze dosáhnout v souřadnicovém systému, kde osa x je zarovnána se směrem vektoru od vysílače k přijímači. Transformace souřadnic se dále provádí otáčením os x a y o úhel rovnající se *point_theta*. Parametry rozdílového vektoru v nové souřadnicové soustavě jsou uloženy v proměnných *rot1_x*, *rot1_y* a *rot1_z*.

Vyzařování vysílací antény musí směřovat směrem k přijímači. To vyžaduje rotaci antény o rozdíl mezi těmito úhly. Nejdříve Θ a potom Φ. Obdobně z hlediska rozdílového vektoru lze na tuto rotaci pohlížet opačně, nejdříve Φ a potom Θ. Souřadnice nového rozdílového vektoru jsou uloženy v proměnných *rot2_x*, *rot2_y* a *rot2_z*.

Počáteční transformace souřadnic, která vedla k jednoduššímu výpočtu rotace antény, může být nyní odstraněna. Výsledné parametry rozdílového vektoru jsou uloženy v proměnných *rot3_x*, *rot3_y* a *rot3_z*.

Výsledné souřadnice rozdílového vektoru slouží k vyhodnocení úhlů Φ a Θ, které určují směr přijímače s ohledem na polohu vysílací antény.

Nejprve je vypočtena vzdálenost mezi vysílačem a přijímačem v rovině xy. Je-li tato vzdálenost 0, vysílač je umístěn buď přímo v přijímači, nebo nad přijímačem. V tomto případě, je-li rozdíl hodnot z záporný, vysílač je umístěn nad přijímačem a svislá rovina úhlu Φ od vysílače do přijímače je -90^o. Jestliže je rozdíl z kladný, Φ je 90^o. Jestliže je vzdálenost vysílače a přijímače v rovině xy větší než nula, úhel Φ k vertikální rovině je tangenta úseku z dělená vzdáleností v rovině xy, jak je zobrazeno na obr. 3.9. [7]

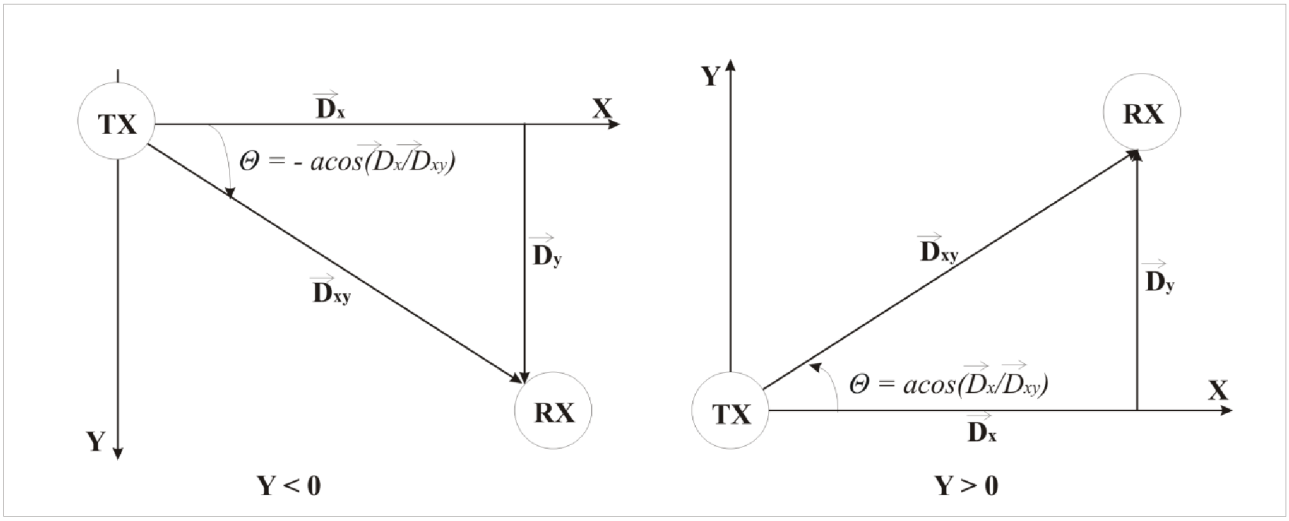


Obr. 3.9 Výpočet Φ

Obdobně jako v předchozím případě, jestliže je souřadnice *y* rozdílového vektoru $\vec{D}$ kladná, úhel Θ v rovině *xy* se vypočítá podle vzorce (3.2), jak ukazuje obr. 3.10. Jestliže je souřadnice rozdílového vektoru $\vec{D}$ záporná, úhel Θ v rovině *xy* se vypočítá podle vzorce (3.3), jak ukazuje obr. 3.10.

$$\Theta_{xy} = -arccos\left(\frac{D_x}{D_{xy}}\right) \tag{3.2}$$

$$\Theta_{xy} = -arccos\left(\frac{D_x}{D_{xy}}\right) \tag{3.3}$$



Obr. 3.10 Výpočet Θ

Rozsah pohybu vysílací antény po rovině úhlu Φ je 0^o kolmo vzhůru až 180^o kolmo dolů, zatímco v následujících krocích je vyžadován rozsah 90^o až -90^o.

5. KROK. ZPOŽDĚNÍ VYSÍLANÍ

Krok „Zpoždění vysílaní“ je šestý krok přenosu, a je určen atributem vysílače *propdel model*. Je volán pro každý přijímač, který úspěšně prošel kroky „Rezervace linky“ a „Výběr kanálu“. Cílem tohoto kroku je výpočet času potřebného pro cestu paketu z vysílače na přijímač. Tento výsledek je obecně závislý na vzdálenosti mezi zdrojem a cílem. Jádro používá tento výsledek k naplánování začátku příjmu pro přijímač, kterému je paket určen. Kromě toho je výsledek zpoždění vysílání v souvislosti s výsledkem kroku „Zpoždění přenosu“.[7]

Volání modelu:

Tento model – model kroku „Zpoždění vysílaní“ se jmenuje *dra_propdel*. Zdrojový kód je k dispozici v souboru *dra_propdel.ps.c* (příloha g), který se nachází v adresáři */models/std/wireless*.

Výpočet zpoždění vysílání se provádí nezávisle pro každý paket přenosu, který úspěšně projde předchozími kroky. Zpoždění vysílání je čas mezi vysláním prvního bitu na vysílači a

přijetím prvního bitu na přijímači. Simulační jádro provádí výpočet ihned po dokončení předchozího kroku bez simulovaného zpoždění a čas volání je počáteční čas přenosu paketu.

Jelikož se vzdálenost mezi vysílačem a přijímačem může u různých paketů měnit v závislosti na přijímacím uzlu, model počítá dva výsledky. Prvním je výpočet zpoždění na začátku vyslání paketu a druhý je výpočet zpoždění na konci přenosu paketu. Simulační jádro očekává výsledky uložené v proměnných TDA *OPC_TDA_RA_START_PROPDEL* a *OPC_TDA_RA_END_PROPDEL*. [7]

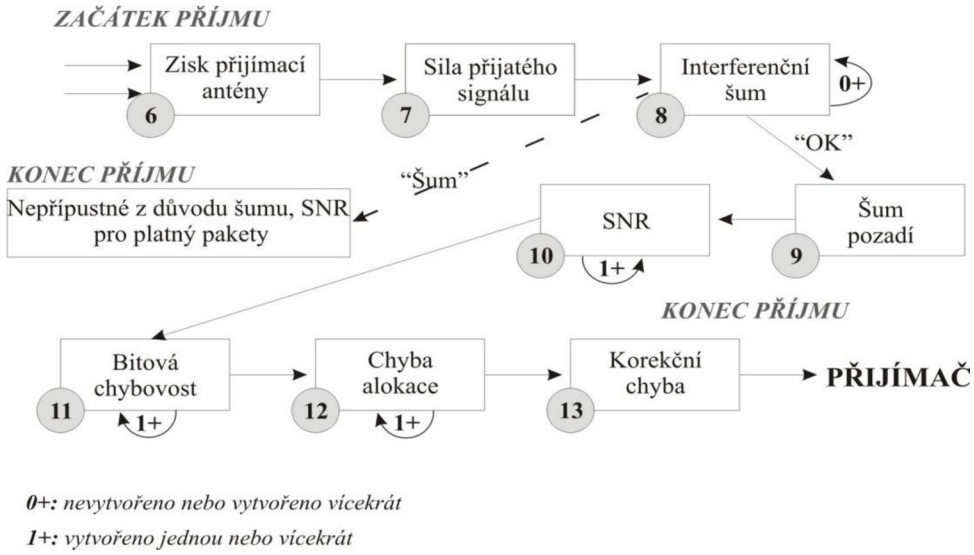
Výpočet zpoždění vysílání

Model zpoždění vysílání počítá zpoždění v důsledku šíření radiového vlnění na základě vzdálenosti oddělující vysílač a přijímač a rychlosti šíření radiových vln.

Model získá vzdálenosti oddělující vysílač a přijímač na začátku a na konci přenosu z TDA *OPC_TDA_RA_START_DIST* a *OPC_TDA_RA_END_DIST* poskytované simulačním jádrem. Oba výsledky zpoždění mohou být získány vydělením těchto vzdáleností rychlostí šíření signálu, která je definována v TDA *PROP_VELOCITY*. [7]

3.3 Popis kroků a výchozích modelů přijímače

Na obr. 3.11 jsou zobrazeny kroky radiového přenosu přijímačů v prostředí OM.

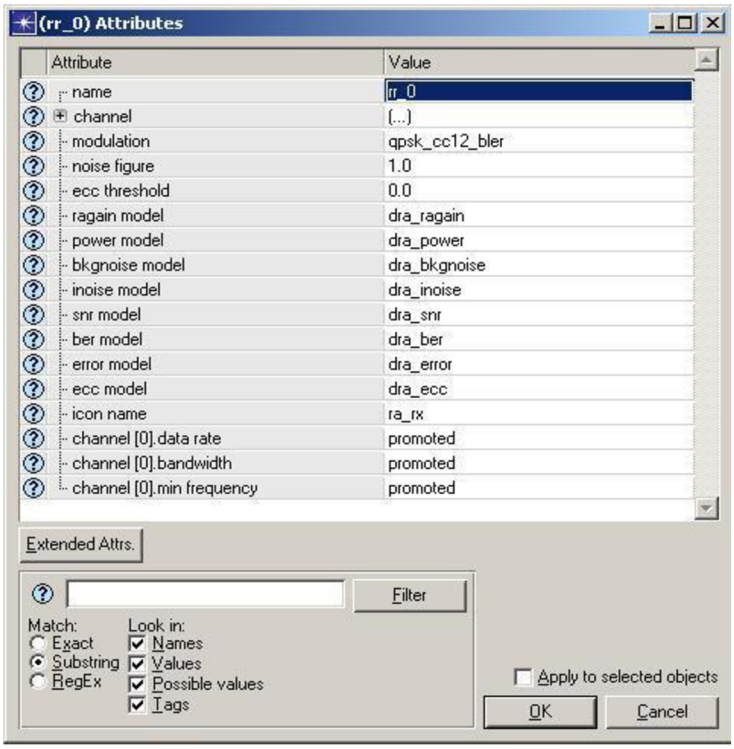


0+: nevytvořeno nebo vytvořeno vícekrát

1+: vytvořeno jednou nebo vícekrát

Obr. 3.11 Kroky radiového přenosu přijímačů v prostředí OM

Na obr. 3.12 je ukázán příklad nastavení kroků přenosu přijímačů v prostředí OM.



Obr. 3.12 Nastavení kroků přenosu přijímačů v prostředí OM

6. KROK. ZISK PŘIJÍMACÍ ANTÉNY

Krok „Zisk přijímací antény“ je sedmý krok přenosu. Jedná se o krok týkající se strany přijímač, protože je určen atributem přijímače *ragain model*.

Je vytvořen samostatně pro každý cílový kanál a jeho volání probíhá v době, kdy náběžná hrana paketu dorazí na místo přijímače.

Účelem kroku je výpočet zisku poskytovaného příslušné anténě přijímače, založeného na směru vektoru vedoucího od přijímače k vysílači. Simulační jádro samo o sobě tento výsledek nevyužívá, ale je typickým faktorem při výpočtu 7. kroku.

Pojetí zisku přijímací antény je stejné jako zisk vysílací antény, nejen z důvodu fyzické konfigurace a realizace antény spojené s přijímačem. [7]

Volání modelu:

Tento model – model krou „Zisk přijímací antény“ se jmenuje *dra_ragain*. Zdrojový kód je k dispozici v souboru *dra_ragain.ps.c* (příloha h), který se nachází v adresáři */models/std/wireless*.

Model určuje zisk přijímací antény definovaný na paketu. Volání logicky navazuje na volání modelu zpoždění vysílání a čas volání je časem příjmu paketu podle předchozího modelu.

Výsledek kroku volaného pro každý paket je číslo s plovoucí desetinnou čárkou, jež představuje zisk antény příslušného přijímacího uzlu. Tato hodnota je vyjádřena v dB a simulační jádro očekává, že tato hodnota bude uložena pro každý paket v TDA *OPC_TDA_RA_RX_GAIN*. [7]

Výpočet zisku přijímací antény

Zisk přijímací antény se počítá obdobně jako zisk vysílací antény. Rozdíl je pouze v tom, že vstupní parametry pro výpočet jsou načteny z TDA souvisejících s přijímačem. Např. *OPC_TDA_RA_RX_PATTERN* namísto *OPC_TDA_RA_TX_PATTERN*. Kromě toho jsou výpočty vztaženy na umístění přijímače ve stejném souřadnicovém systému jako v případě vysílače. Jelikož je výpočet dále stejný, lze k popisu využít popis modelu *dra_tagain*. [7]

7. KROK. SÍLA PŘIJATÉHO SIGNÁLU

Krok „Síla přijatého signálu“ je osmý krok přenosu a je určen atributem přijímače *power model*. Je prováděn samostatně pro každý cílový kanál. Cílem tohoto kroku je výpočet přijaté energie přijímaných paketů ve Wattech.

Pro přijaté pakety, které jsou klasifikovány jako platné[1], je výsledek energie klíčovým faktorem při určování schopnosti přijímače správně zachytit informace v paketu. U paketů, které jsou klasifikovány jako „šum“, musí být přijímaný výkon obvykle stále hodnocený pro podporu výpočtu poměru intenzity platných a šumových paketů.

Obecně platí, že výpočet přijaté energie je založen na faktorech, jako je výkon vysílače, vzdálenost mezi vysílačem a přijímačem, vysílací frekvence a zisk vysílací a přijímací antény. [7]

[1] Platné jsou ty pakety, které splňují kritéria pro správný přenos kanálu. Obecně platí, že je nutné, aby kanály přijímače a vysílače měly stejné vlastnosti a také to, že kanál přijímače je možné synchronizovat do paketů signálu.

$$L_p = \left(\frac{\lambda}{4\pi D}\right)^2, \tag{3.4}$$

kde

L_p [-] je útlum signálu;
λ[m] je vlnová délka;
D [m] je vzdálenost.

$$\lambda = \frac{c}{f_c}, \tag{3.5}$$

kde

c[m/s] je rychlost světla;
f_c [Hz] je střední frekvence.

$$P_i = \frac{P_{tx}(f_{max} - f_{min})}{B}, \tag{3.6}$$

kde

P_i [W] je výkon vysilače;
f_{max} [Hz] je *maximální frekvence;*
f_{min} [Hz] je minimální frekvence;
B [Hz] je šířka pásma.

$$P_{rx} = P_i\, G_{tx} L_p G_{rx}, \tag{3.7}$$

kde

P_{tx} [W] je výkon přijaté energie;
G_{tx} [W] je zisk antény vysilače;
G_{rx} [W] je zisk antény přijímače.

Volání modelu:

Tento model – model kroku „Síla přijatého signálu“ se jmenuje *dra_power*. Zdrojový kód je k dispozici v souboru *dra_power.ps.c* (příloha i), který se nachází v adresáři */models/std/wireless*. Model počítá průměrný výkon úrovně přijatých signálů v přijímacím kanálu. Tento krok umožňuje výpočty v následujících modelech přijímacího kanálu jako SNR a BER.

Výpočet je prováděn nezávisle pro každý paket, který tohoto kroku dosáhne. Model je simulačním jádrem volán ihned po dokončení předchozího kroku bez simulovaného zpoždění. Volání probíhá ihned po začátku příjmu paketu. Tento čas je dán součtem času zahájení přenosu a zpoždění vysílání a je k dispozici v TDA *OPC_TDA_RA_START_RX*.

Výsledek kroku volaného pro každý paket je 64 bitové číslo s plovoucí desetinnou čárkou, jež představuje sílu přijatého signálu. Simulační jádro očekává, že tato hodnota bude uložena pro každý paket v TDA *OPC_TDA_RA_RCVD_POWER*. [7]

Výlučnost příjmu

Součástí modelu je možnost uzamčení přijímacího kanálu z důvodu současného příjmu více paketů najednou. Přestože model umožňuje souběžný příjem, není to obvyklé a je tomu zabráněno speciální funkcí umístěnou v modelech kroků 7 a 13. Tato funkce není hlavním cílem těchto modelů, ale byla do nich začleněna.

Uzavření přijímacího kanálu je reprezentováno jako bitová hodnota „*obsazeno*“ nebo „*volno*“. Hodnota *OPC_TRUE* indikuje, že přijímací kanál momentálně zpracovává platný paket. *OPC_FALSE* indikuje stav přijímacího kanálu, kdy nezpracovává buď žádný paket, nebo paket, který byl ve třetím kroku přenosu označen jako neplatný.

Model *dra_power* se používá k ověření stavu přijímacího kanálu, protože je volán přesně v okamžiku příjmu paketů. Pouze platné pakety přicházející do přijímacího kanálu mohou způsobit zaneprázdnění paketu. Pokud dorazí neplatný paket, kanál je volný, nicméně výpočet v modelu probíhá, jelikož jeho šum může mít vliv na správný příjem následujících paketů.

V této části jsou pakety testovány na hodnotu TDA *OPC_TDA_RA_MATCH_STATUS*. Platné pakety budou mít ve třetím kroku přenosu přiřazenu hodnotu *OPC_TDA_RA_MATCH_VALID*. Model dále získá ID objektu přijímacího kanálu, na kterém je paket testován. ID je přiřazeno simulačním jádrem a je uloženo v TD*A* *OPC_TDA_RA_RX_CH_OBJID*. Toto číslo je použito v informaci o zaneprázdnění přijímacího kanálu.

Je-li již přijímací kanál zaneprázdněn, nový přicházející paket je označen jako neplatný. To je provedeno nastavením TDA *OPC_TDA_RA_MATCH_STATUS* na hodnotu symbolické konstanty *OPC_TDA_RA_MATCH_NOISE* a tím je zabráněno později přijatému paketu, aby byl korektně přijmut a zpracován z přijímacího kanálu.

Jestliže je však na druhou stranu přijímací kanál zaneprázdněn, není nastaveno (0, *OPC_FALSE*), a přijímací kanál přicházející paket přijme a zůstává nadále v platnosti, tedy hodnota jeho TDA *OPC_TDA_RA_MATCH_STATUS* se nemění. Přijímací kanál je potom uzamčen přiřazením hodnoty *OPC_TRUE*. [7]

Výpočet síly přijatého signálu

Průměrná síla přijímaného signálu pro všechny přicházející pakety (platné i neplatné) je počítána ve zbytku kódu modelu *dra_power*. Výpočet zahrnuje počáteční vysílací výkon, ztrátu výkonu při přenosu radiových vln a zisk vysílací a přijímací antény.

Výkon ve wattech je načten z TDA *OPC_TDA_RA_TX_POWER* a hodnota je uložena do proměnné *tx_power*. Stejným postupem je získána základní frekvence přenosu a šířka pásma z TDA *OPC_TDA_RA_TX_FREQ* a *OPC_TDA_RA_TX_BW*. Hodnoty jsou uloženy do proměnných *tx_base_freq* a *tx_bandwidth*. Hodnoty těchto proměnných dále slouží k výpočtu střední frekvence přenosu dat, která je uložena do proměnné *tx_center_freq*.

Dráha šíření vlnění v metrech je získána z TDA *OPC_TDA_RA_START_DIST* a hodnota uložena v proměnné *prop_distance*. Je-li využíváno terénní modelování TMM, hodnota ztrát při šíření vlnění je již definována. Pokud ne, ztráty jsou vypočítány ze závislosti na vlnové délce podle vztahu 3.4. [7]

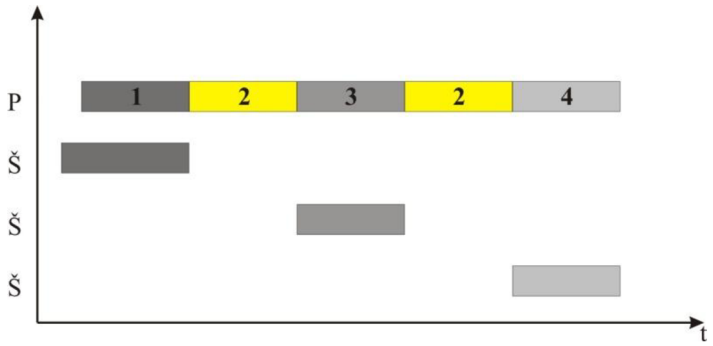
Výsledek výpočtu je uložen do proměnné *path_loss*. V případě, že vzdálenost mezi vysílačem a přijímačem je nulová, ke ztrátám při šíření vlnění nedochází a proměnná *path_loss = 1*.

Nakonec jsou pro výpočet načteny zisky vysílací a přijímací antény z TDA *OPC_TDA_RA_TX_GAIN* a *OPC_TDA_RA_RX_GAIN*. Načtené hodnoty v dB jsou převedeny do nelogaritmických hodnot, aby jimi bylo možné provádět matematické operace s dalšími proměnnými. Proměnná *rcvd_power* je potom vypočtena jako součin *tx_power*, *tx_ant_gain*, *path_loss a rx_ant_gain*. Tento výsledek je přiřazen k TDA *OPC_TDA_RA_RCVD_POWER* pro použití v následujících krocích přenosu. [7]

8. KROK. INTERFERENČNÍ ŠUM

Krok „Interferenční šum“ je devátý krok přenosu a je určen atributem přijímače *inoise model*. Krok lze pro paket uplatnit ve dvou případech: paket je platný a dorazí na místo určení, zatímco jiný paket byl již přijat, anebo paket je platný a již dorazil, když jiný paket (platný nebo neplatný) přichází. Je zřejmé, že k první okolnosti dochází maximálně jednou pro každý paket a ke druhé může dojít vícekrát v závislosti na přenosové činnosti ostatních vysílačů v modelu. Platí, že jedno vyvolání tohoto kroku může být sdíleno s přenosem dvou paketů, pokud jsou oba platné.[7]

Cílem tohoto kroku je popis vzájemného působení mezi přenosy, které přichází souběžně na stejné přijímače. Simulační jádro vyhrazuje TDA reprezentovaný symbolickou konstantou *OPC_TDA_RA_NOISE_ACCUM* za účelem uložení aktuální úrovně šumu ze všech rušivých vysílání. Tento zásobník je zachováván pouze pro platné pakety, protože obecně není třeba hodnotit kvalitu spojení šumu. Tudíž tento krok předpokládá zvýšit hodnotu tohoto zásobníku při každém platném paketu obdrženém o výkonu rušivých paketů. Když paket (platný nebo neplatný) doplňuje příjem, jádro automaticky odečte jeho příjmový výkon od hodnoty šumu zásobníku všech platných paketů, které stále přicházejí na kanál. Tímto způsobem zásobník odráží pouze aktuální hladinu šumu (obr. 3.13). [7]



Obr. 3.13 Varianta interferenčního šumu

Volání modelu:

Tento model – model kroku „Interferenční šum“ se jmenuje *dra_inoise*. Zdrojový kód je k dispozici v souboru *dra_inoise.ps.c* (příloha j), který se nachází v adresáři */models/std/wireless*.

Model definuje rušení, které vzniká současným působením více přenosů současně. Skutečný dopad tohoto rušení závisí na vlastnostech vysílače a přijímače a časování vysílání. Tento model provádí analýzu těchto vlastností a určuje míru vzájemného rušení souběžných přenosů.

K výpočtu interferenčního rušení dochází pouze v případě, když dva pakety dorazí souběžně na jeden radiový přijímač. Pokud při přenosu nedojde ke kolizi paketů, tento krok nemusí být volán. Naopak může být model během přenosu volán i několikrát, pokud dojde ke kolizi s více než jedním paketem. Volání modelu probíhá vždy s příchodem platného paketů do přijímače, pokud je již jiný paket zpracováván, nebo s příchodem nového paketu, pokud je již zpracováván jiný platný paket. Příchod druhého paketu způsobí kolizi. Čas volání modelu je čas příchodu druhého paketu.

Výsledkem modelu jsou dvě 64 bitové čísla s plovoucí desetinnou čárkou, jež představují rušivý výkon ovlivňující sousední paket. Simulační jádro očekává, že tyto hodnoty budou uloženy pro každý paket zvlášť v TDA *OPC_TDA_RA_NOISE_ACCUM*. [7]

Výpočet interferenčního šumu:

Výchozí model *dra_inoise* filtruje pakety, které se přímo překrývají testováním ukončení příjmu dříve přijatého paketu vůči času volání. Pakety, které dokončí příjem současně s časem volání modelu, mohou kolizi uniknout, protože ve skutečnosti je překrytí paketů reálným výsledkem libovolného pořadí událostí, které se vyskytují ve stejný čas simulace.

Jestliže došlo k přijetí nového paketu ještě před ukončením příjmu již zpracovávaného paketu, model zvýší počet kolizí oběma paketům a zjistí, zda se jedná o platné nebo neplatné pakety a načte sílu signálu obou paketů. Výpočet šumu je nutný pouze pro ovlivněné platné pakety, jelikož neplatné pakety nejsou určeny k příjmu a jejich kvalita není důležitá. Pro platné pakety je podíl šumu založen na síle signálu interferujících paketů.

Simulační jádro automaticky odečte průměrný přijatý výkon šumu dle atributu akumulovaného rušení, jakmile interferenční pakety dokončí přenos. Důsledkem tohoto chování je časově proměnná úroveň interferenčního šumu a tedy časově proměnný poměr SNR a bitová chybovost BER. Tyto veličiny jsou počítány v následujících krocích 10 a 11. [7]

9. KROK. ŠUM POZADÍ

Krok „Šum pozadí“ je desátý krok přenosu a je určen atributem přijímače *bkgnoise model*. Cílem tohoto kroku je reprezentovat účinek všech zdrojů šumu s výjimkou jiných současně přicházejících přenosů. Očekávaným výsledkem je součet výkonů ve Wattech jiných zdrojů šumu, měřených v místě přijímače. Typické zdroje šumu pozadí jsou tepelný nebo kosmický šum, vyzařování elektroniky a jiné nedefinované zdroje záření (např. rádio, amatérské rádio nebo televize v závislosti na frekvenci) [7].

Volání modelu:

Tento model se jmenuje *dra_bkgnoise*. Zdrojový kód je k dispozici v souboru *dra_bkgnoise.ps.c* (příloha k), který se nachází v adresáři */models/std/wireless*.

Model umožňuje výpočet šumu, jehož zdrojem nejsou jiné vysílací moduly. Jejich zdrojem může být například reliktní záření, tepelné záření nebo městské elektromagnetické znečištění v závislosti na tom, kde se přijímač nachází.

Výsledek kroku je 64 bitové číslo s plovoucí desetinnou čárkou, jež představuje sumu šumu těchto zdrojů. Tato hodnota bude uvažována při výpočtu SNR v následujícím modelu. Simulační jádro očekává, že tato hodnota bude uložena v TDA *OPC_TDA_RA_BKGNOISE*. [7]

Výpočet šumu pozadí.

Výchozí model *dra_bkgnoise* při výpočtu šumu pozadí uvažuje vliv konstantní hodnoty okolního šumu, šumu pozadí a tepelného šumu na přijímač.

V rámci okolního šumu může být uvažována spektrální hustota radiového znečištění městského prostředí a v modelu *dra_bkgnoise* je tato hodnota definována symbolickou konstantou *AMB_NOISE_LEVEL*. Tato hodnota může být změněna v závislosti na modelové situaci.

Výsledný šum pozadí je charakterizován efektivní teplotou pozadí, která se přidává k efektivní teplotě zařízení přijímače. Teplota pozadí je definována symbolickou konstantou *BKG_TEM*. Šumové číslo přijímače je definováno za předpokladu provozní teploty 290K. Součet těchto teplot, z nichž každá je zdrojem jiného šumu, se vynásobí šířkou pásma přijímacího kanálu a Boltzmannovou konstantou a výsledná hodnota se přidá ke zpracovávanému signálu, jak je uvedeno ve vzorci (3.8).

$$P_{bkg} = \left(T_{rx} + T_{bkg}\right) \cdot \Delta f \cdot k, \tag{3.8}$$

kde

P_{bkg} *[W] je výkon šumu pozadí,*
T_{rx} *[K] je teplota přijímače,*
T_{bkg} *[K] teplota pozadí,* $T_{bkg} = 290$,
Δf *[Hz] je šířka pasma přijímacího kanálu,*
k *[J/K] je Boltzmannova konstanta,* $k = 1{,}379 \cdot 10^{-23}$.

Tento šum je přidán k okolnímu šumu pro modelování celkového účinku šumu. Zesilovač přijímače v tomto modelu není uvažován. [7]

10. KROK. POMĚR SIGNÁL-ŠUM (SNR)

Krok „Poměr signál-šum (SNR)“ je jedenáctým krokem přenosu a je určen atributem přijímače *snr model*.

Ten lze uplatnit na platný paket (jak je stanoveno v 3. kroku) ve třech případech:

- paket dorazí na místo určení kanálu;
- paket je již přijat a další paket (platný nebo neplatný) přichází;
- paket je již přijat a další paket (platný nebo neplatný) dokončil příjem.

Je zřejmé, že první okolnost nastane právě jednou pro každý paket a druhá a třetí může být opakována v závislosti na přenosové činnosti ostatních vysílačů v modelu. Tyto tři typy volání definují interval, během kterého se průměrný výkon SNR paketů bere jako konstantní (jedná se o aproximaci, kde je třeba očekávat proměnlivá data, protože SNR se bude průběžně měnit).[7]

Cílem kroku „SNR“ je výpočet aktuálního průměru výkonu SNR pro přicházející paket. Tento výpočet je obvykle založen na hodnotách získaných při dřívějších krocích, včetně přijaté energie, šumu pozadí a interferenčního šumu. Pro výpočet SNR paketu je důležité měření výkonu,

které pomáhá stanovit schopnost přijímače správně přijímat obsah paketů. Výsledek získaný v tomto kroku jádro používá k aktualizaci výsledků standardního výstupu přijímače a obvykle i pro pozdější kroky přenosu. [7]

Volání modelu:

Tento model – model kroklu „Poměr signál-šum (SNR)“ se jmenuje *dra_snr*. Zdrojový kód je k dispozici v souboru *dra_snr.ps.c* (příloha l), který se nachází v adresáři */models/std/wireless*.

Výpočet SNR závisí na výpočtu šumu pozadí a interferenčním šumu z jiných přenosů. Zatímco šum pozadí je vyhodnocován během zpracování paketu pouze jednou na začátku příjmu paketu, zdroje interferenčního rušení se mohou během příjmu paketu měnit. Proto i SNR je během přenosu třeba hodnotit pro daný paket vícekrát. Část paketu, která se na přijímači zpracuje mezi jednotlivými aktualizacemi SNR, se nazývá segment paketu. V jednom segmentu je úroveň SNR konstantní.

Výsledkem kroku jsou dvě 64 bitová čísla s plovoucí desetinnou čárkou, jež představují podíl středního výkonu signálu k výkonu šumu pro přijímaný paket v dB a čas, ve kterém bylo SNR počítáno. Simulační jádro očekává, že tyto hodnoty budou uloženy v TDA *OPC_TDA_RA_SNR* a *OPC_TDA_RA_SNR_CALC_TIME*. [7]

Výpočet SNR

SNR počítáno v tomto kroku je poměr středního výkonu přijímaného signálu k průměrnému akumulovanému výkonu všech zdrojů šumu pozadí a interferenčního šumu. Tyto hodnoty byly vypočteny v předchozích krocích přenosového kanálu, a proto jsou k dispozici v rámci TDA. Výchozí model *dra_snr* počítá tento poměr tak, že načte průměrné hodnoty šumu pozadí a interferenčního šumu z TDA a jejich součtem podělí sílu přijímaného signálu. Podle požadavků simulačního jádra je výsledek v dB. [7]

11. KROK. BITOVÁ CHYBOVOST (BER)

Krok „Bitová chybovost (BER)“ je dvanáctý krok přenosu a je určen atributem přijímače *ber model*.

Ten může být uplatněn na platný paket (jak je stanoveno v 3. kroku ve třech případech:

- paket dorazí na místo určení kanálu;
- paket je již přijat a další paket (platný nebo neplatný) přichází;
- paket je již přijat a další paket (platný nebo neplatný) doplňuje příjem.

Tyto okolnosti odpovídají konečné fázi, během které je SNR paketu uvažován jako konstanta.

Cílem kroku „BER“ je odvození pravděpodobnosti bitových chyb v aktuálním interval s konstantním SNR. Nejedná se empirickou bitovou chybovost, ale o očekávaný poměr, obvykle založený na SNR. Obecně platí, že chybovost u tohoto kroku je také funkcí použitého typu modulace pro vysílání signálu.[7]

Volání modelu:

Tento model – model kroku „Bitová chybovost (BER)“ se jmenuje *dra_ber*. Zdrojový kód je k dispozici v souboru dra_ber.ps.c (příloha m), který se nachází v adresáři */models/std/wireless*.

Bitová chybovost BER je typickou funkcí pro poměr signál-šum SNR. Stejně jako SNR, tak i BER je počítána ne na celém paketu, ale na jeho segmentech, kde jsou definovány změny SNR. BER je počítána zpětně v čase, kdy segment paketu končí, protože obvykle není možné průběh SNR předvídat. Na posledním segmentu paketu se BER měří v době dokončení příjmu paketu.

Výsledek kroku je 64 bitové číslo s plovoucí desetinnou čárkou, jež představuje očekávanou hodnotu bitové chybovosti BER na segmentu paketu. Simulační jádro očekává, že tato hodnota bude uložena v TDA *OPC_TDA_RA_BER*. [7]

Výpočet BER.

Výchozí model pro výpočet BER se nazývá *dra_ber*. Tento model hodnotí BER na základě předem vypočítané hodnoty SNR a také odpovídá za zpracování zisku přijímače. SNR v dB je načtena z TDA *OPC_TDA_RA_SNR*. Tato hodnota je uvažována do zpracování zisku také v dB pro získání efektivní hodnoty SNR. Efektivní SNR může být převeden z logaritmické stupnice a vyjádřen jako poměr Eb/N0, kde Eb je přijatá energie bitu v (J) a N0 je spektrální hustota výkonu šumu ve (W/Hz). BER je odvozen od efektivního SNR založeného na modulační křivce spojené s přijímacím kanálem. Tato funkce, která definuje BER pro každou hodnotu efektivního SNR, může být specifikována v **Antenna Curve Editor** nebo **Modulation Curve Editor**. [7]

12. KROK. ALOKACE CHYB

Krok „Alokace chyb“ je třináctý krok přenosu a je určen atributem přijímače *error model.*

Je proveden vždy ihned po opuštění kroku BER. Cílem kroku „alokace chyb“ je odhad počtu bitových chyb v segmentu paketů, kde pravděpodobnost bitové chyby byla vypočítána a je konstantní. Jestliže se v průběhu příjmu paketu nevyskytují změny v pravděpodobnosti bitové chyby, tento segment by mohl být celý paket. Odhad počtu bitových chyb je obvykle založen na pravděpodobnosti bitových chyb získané z kroku 11 a délce sledovaného segmentu. [7]

Volání modelu:

Tento model – model kroku „Alokace chyb“ se jmenuje *dra_error*. Zdrojový kód je k dispozici v souboru *dra_error.ps.c* (příloha n), který se nachází v adresáři */models/std/wireless*.

Pakety přenášené radiovým spojením mohou být ovlivněny interferenčním šumem i šumem pozadí. Tyto zdroje rušení mají vliv na kvalitu příjmu, který lze charakterizovat průměrnou bitovou chybovostí BER. BER bylo vypočítáno v předchozím kroku. Úkolem modelu alokace chyb je použití daného BER pro vyhledání chyb přenosu v daném paketu.

Vzhledem k tomu, že BER stejně jako SNR, jak bylo dříve uvedeno, se aplikuje na segmentech paketů, tak i model alokace chyb je volán pro každý segment paketu zvlášť, kde SNR zůstává konstantní. Simulační jádro očekává, že teto model vypočítá počet bitových chyb u každého segmentu paketu a přidá tuto hodnotu k počtu bitových chyb postihujících celý paket. Zásobník bitových chyb celého paketu, kde se tato hodnota ukládá má v TDA název

OPC_TDA_RA_NUM_ERRORS. Kromě toho simulační jádro očekává, že model vypočítává okamžitou bitovou chybovost postihující segmenty paketů a tuto hodnotu ukládá v TDA *OPC_TDA_RA_ACTUAL_BER*. [7]

Algoritmus alokace chyb

Algoritmus modelu využívá hodnoty BER. Vzhledem k této bitové chybovosti a počtu bitů v segmentech paketu model určí celkový počet bitových chyb. Před vysvětlením algoritmu je vhodné zmínit alternativní metodu, která je jednoduchá a přesná, ale neefektivní.

Nejreálnější model pro stanovení počtu bitových chyb v daném paketu je napodobení chování přijímače, který rozhoduje o správnosti bitu na základě pravděpodobnosti chyby. Tento model prochází každý bit paketu a na základě rovnoměrného rozložení náhodných čísel rozhodne o chybě bitu.

Jelikož tento algoritmus testuje každý bit, tak je velmi přesný. Kromě stanovení celkového počtu chyb je navíc schopen poskytnout informace o rozložení chybných bitů. Je však zřejmé, že čas potřebný k provedení algoritmu roste úměrně s délkou paketu. Kromě toho musí být testován každý bit i v systémech s nízkou mírou chybovosti. Algoritmus tedy není dostatečně efektivní pro široký rozsah tříd modelovaných systémů a není vhodný pro použití ve výchozím modelu.

Cílem modelu *dra_error* je vyhnout se postupnému testování každého bitu v paketu a pouze generovat informace o bitových chybách, ale přitom stále přesně vypočítat celkový počet bitových chyb. Vzhledem k efektivnosti modelu by to mělo být realizováno minimálním počtem výpočtů.

Algoritmus vychází z pravděpodobnosti *k* chybných bitů vyskytujících se v segmentu délky *N*. Tato pravděpodobnost se označuje jako *Pk*. Pravděpodobnost, že chybou není postiženo *N* následujících bitů, je označena jako *(1 - p)* a jednána následujícím výrazem (3.9).

$$P_0 = (1-p)^N \tag{3.9}$$

Pravděpodobnost, že právě jedna chyba se vyskytuje v počtu rozdílně uspořádaných bitů v segmentu, je dána následujícím výrazem (3.10):

$$P_1 = p \cdot (1-p)^{N-1} \cdot \binom{N}{1} \tag{3.10}$$

Obecně tedy platí (3.11):

$$P_k = p^k \cdot (1-p)^{N-k} \cdot \binom{N}{k} \tag{3.11}$$

Algoritmus používaný v modelu *dra_error* pro výpočet počtu bitových chyb používá výše uvedený vztah. Algoritmus nejprve generuje náhodné číslo v intervalu 0 – 1, aby mohla být tato hodnota testována s pravděpodobností výskytu různého počtu bitových chyb. Dále začíná iterace. První pravděpodobnost, že došlo k 0 chybám, je počítána podle vztahu výše a porovnána s náhodně generovaným číslem. Je-li náhodné číslo nižší než tato pravděpodobnost, pak je v paketu 0 bitových chyb. Jinak se znovu testuje pravděpodobnost výskytu 1 nebo méně chyb a tak dále. Je-li náhodné číslo menší než tato pravděpodobnost, přesto vyšší než pravděpodobnost výskytu z předchozího počtu bitových chyb, potom počet bitových chyb v paketu je 1. Tímto způsobem algoritmus

iterativně pokračuje, dokud hodnota bitových chyb *k* nebo méně je větší než počáteční náhodné číslo, tak počet chyb v paketu je *k*. [7]

Realizace modelu

Model určuje počet bitových chyb v paketu na základě bitové chybovosti a délky paketu. Proto je nejprve načtena hodnota bitové chybovosti BER a délka paketu. BER je načteno z TDA *OPC_TDA_RA_BER* a délka paketu se získá procedurou jádra *op_pk_total_size_get()*.

Před výkonnou částí algoritmu se mohou vyskytnout speciální případy, které přidělují přesný počet bitových chyb, tedy BER = 0,0 nebo 1,0. Jestliže je BER 0,0, lze počet chyb paketu určit ihned také jako nulový bez nutnosti dalších výpočtů. Obdobně v případě, když BER = 1,0, lze počet chyb určit jako počet bitů v paketu.

Ostatní případy jsou zpracovávány iterativním algoritmem popsaným výše. [7]

13. KROK. KOREKCE CHYB

Krok „Korekce chyb“ je čtrnáctý a závěrečný krok přenosu a je určen atributem přijímače *ecc model*. Alternativní verze tohoto modelu pro sítě nepoužívající informace o stavu přijímaného kanálu se nachází v souboru *dra_ecc_no_rxstate.ps.c*.

Pro každý platný paket, určený krokem 3 „výběr kanálu“, dojde k jedinému volání tohoto kroku. Cílem kroku je určit, zda může být přicházející paket přijat a předán prostřednictvím kanálové korespondence výstupního toku dat na jeden ze sousedních přijímacích modulů v cílovém uzlu. To obvykle závisí na tom, zda paket prošel kolizemi, tedy výsledku vypočítaného v kroku alokace chyb a schopnosti přijímače odstranit chyby ovlivňující paket (odtud název kroku). Na základě výsledku tohoto kroku jádro buď paket odstraní, nebo dohlédne na doručení do cílového uzlu. Navíc tento výsledek ovlivňuje výsledky chyb a propustnosti získané pro přijímací kanál. [7]

Volání modelu:

Tento model - model kroku „Korekce chyb“ se jmenuje *dra_ecc*. Zdrojový kód je k dispozici v souboru *dra_ecc.ps.c* (příloha o), který se nachází v adresáři */models/std/wireless*.

Model korekce chyb je posledním krokem radiového přenosu a je volán po označení chyb v posledním segmentu paketu. Čas volání je při dokončení příjmu paketu.

Výsledkem modelu korekce chyb paketu je jedna logická hodnota *OPC_TRUE* nebo *OPC_FALSE*, která označuje, zda je přijatý paket v pořádku nebo nikoli. Výsledek *OPC_FALSE* simulačnímu jádru označuje, že paket by měl být odmítnut, zatímco výsledek *OPC_TRUE* označuje, že paket by měl být přijat. Simulační jádro očekává, že tento výsledek bude uložen v TDA *OPC_TDA_RA_PK_ACCEPT*. [7]

Test přijatelnosti paketu

Model určuje, zda může být paket přijat na základě dvou kriterií. První test pracuje s pakety, které nebyly přijaty celé z důvodu odpojení vysílacího uzlu. Zkrácené pakety, neboli *runts*, jsou označeny za nepřijatelné. Druhý test porovnává podíl bitových chyb ve sledovaném paketu s prahovou hodnotou schopnosti opravy chyb přijímače. [7]

4 MODELOVÁNÍ BEZDRÁTOVÉHO PŘENOSU V OM

V této části semestrální práce bude vytvořena bezdrátová síť s pohyblivým rušícím uzlem. Topologie sítě se skládá ze tří uzlů:

- Uzel vysílače vysílá stejnou intenzitou ve všech směrech. Skládá se z modulu generátoru paketů, modulu vysílače radiových vln a z modulu antény.
- Uzel přijímače zjišťuje kvalitu signálu vysílaného ze stacionárního uzlu vysílače. Skládá se z modulu antény, modulu sink procesoru a pomocného procesoru pro práci se směrovou anténou.
- Mobilní rušící uzel vytváří radiový šum. Trajektorie rušícího uzlu probíhá uvnitř i vně dosahu přijímače a zvyšuje nebo snižuje interferenci v přijímači.

Modul antény (obr. 4.1) modeluje zisk fyzické antény na základě hodnot atributů jejích parametrů. Anténa používá izotropní vzor, který má stejný zisk ve všech.

Obr. 4.1 Modul antény

Modul vysílače radiových vln (obr. 4.2) předává pakety na anténu rychlostí 1024 b/s s využitím 100% šířky pásma.

Obr. 4.2 Modul vysílače radiových vln

Modul radiopřijímače (obr. 4.3) sleduje parametry každého přijímaného paketu z důvodu ověření, zda průměrná BER paketu je menší než zadaná mez v atributu přijímače *ecc threshold*. Jestliže je BER příliš vysoká, paket bude odstraněn.

Obr. 4.3 Modul radiopřijímače

Modul procesoru (obr. 4.4) vypočítává potřebné souřadnice, které anténa potřebuje pro zaměření cílového bodu. Jsou jimi zeměpisná délka a šířka a nadmořská výška. Procesor tyto

výpočty provádí užíváním procedur jádra, které převádí polohu uzlu na komplexní souřadnice pro potřeby antény.

Obr. 4.4 Modul procesoru

V této části jsou data vysílána ze stacionárního vysílače do stacionárního přijímače. Tyto objekty jsou propojeny radiovým spojením. Toto spojení závisí na mnoha různých fyzikálních charakteristikách včetně frekvenčního rozsahu, typu modulace, výkonu vysílače, vzdálenosti a směru antény.

4.1 Vytváření modelů uzlů

Pro model bezdrátové sítě bude potřeba vytvořit tři modely uzlů: vysílač, přijímač a rušící uzel.

4.1.1 Vysílací uzel

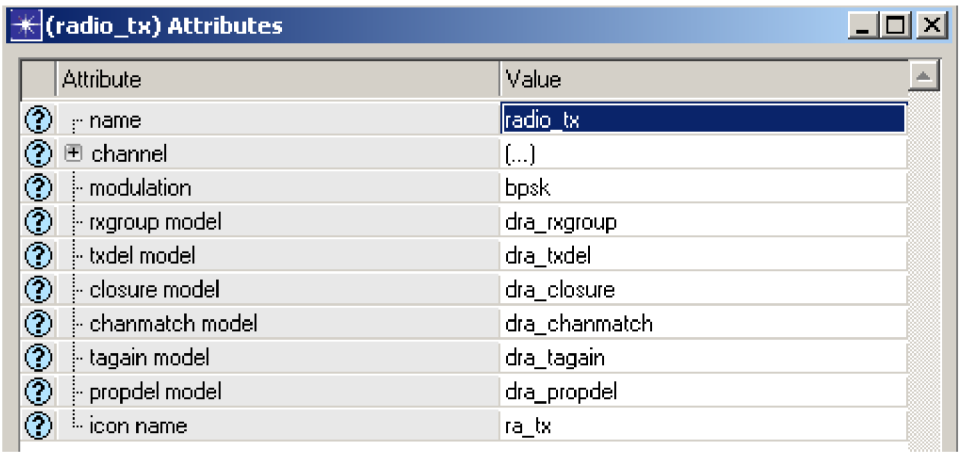
Vysílací uzel (obr. 4.5) sestává z generátoru paketů, modulu bezdrátového vysílače a antény. Generátor paketů generuje pakety o velikosti 1024 bitů s rychlostí 1 paket/s v konstantním intervalu (jedná se o výchozí hodnoty).

Obr. 4.5 Vysílací uzel

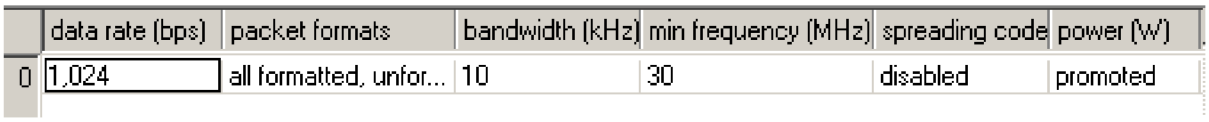
Poté, co jsou pakety vygenerovány, putují do vysílacího modulu, který vysílá pakety na kanálu rychlostí 1024b/s s využitím celé šířky pásma. Pakety potom prochází z vysílače vedle dalších kanálů na anténu.

Izotropní anténa vysílá radiové vlny o stejné intenzitě ve všech směrech.

Ke spuštění simulace je potřeba doplnit atributy výkonu užitého kanálu. Po doplnění údajů je možné je kdykoli měnit během simulace. Po kliknutí pravým tlačítkem myši na modulu *radio_tx* je vybrán *Edit Attributes* (obr. 4.6) a poté nastaven parametr *power* na hodnotu *promote* (obr. 4.7).

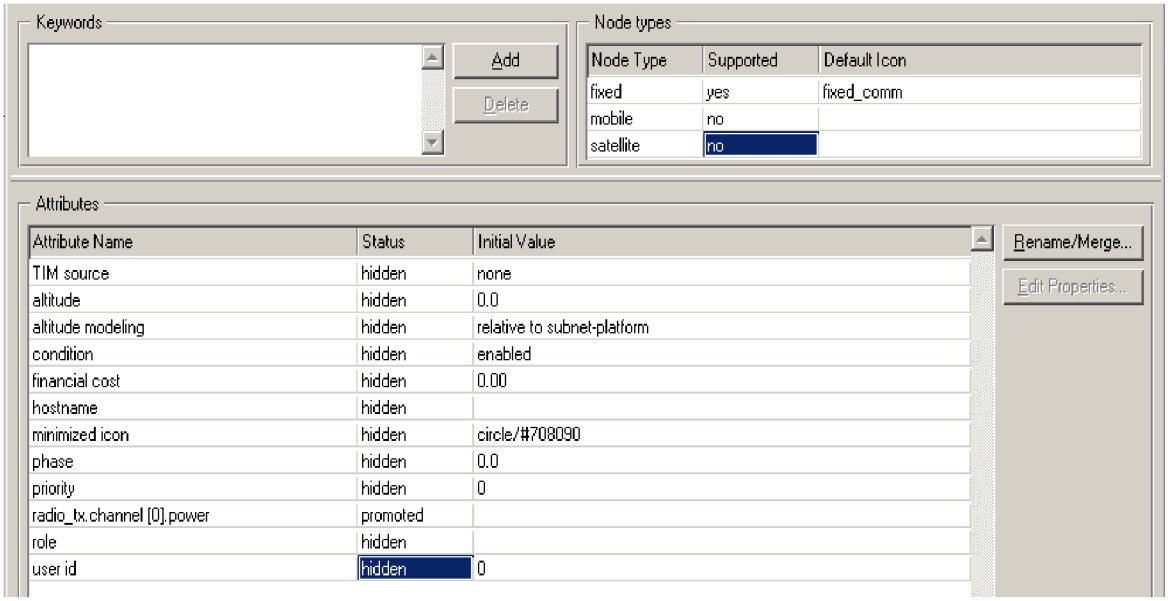


Obr. 4.6 Okno *Edit Attributes* pro vysílací modul



Obr. 4.7 Nastavení parametru *power* na hodnotu *promote*

Nyní budou zadány atributy rozhraní uzlu podle obr. 4.8.

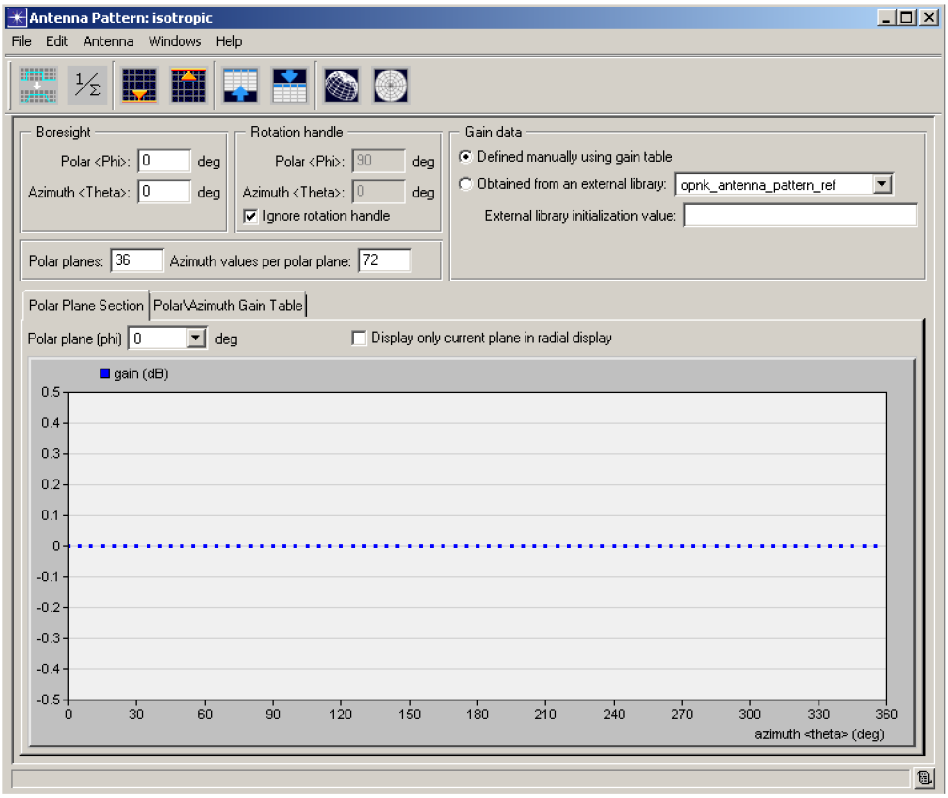


Obr. 4.8 Zadání atributů pro rozhraní uzlu vysílače

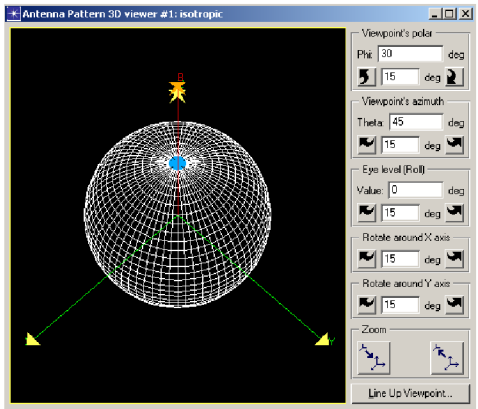
Hodnota atributů se statusem *hidden* bude platná pro všechny další dílčí části uzlu. Hodnota atributu se statusem *promoted* vyjadřuje v další úrovni uzlu vytvoření nového objektu

Editor parametrů antény

Parametry antény se dělí na části pro soustavu polárních souřadnic a azimut. Polární souřadnice představují rovinné horizontální složky soustavy antény. Hodnoty zisku jsou pro každou rovinu určeny příslušným rozsahem azimutu okolo roviny a tak vymezuje vzor antény v horizontální rovině. Budou použity výchozí parametry izotropní antény (obr. 4.9). 3D model antény je na obr. 4.10.



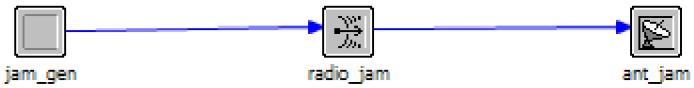
Obr. 4.9 Výchozí parametry antény



Obr. 4.10 3D model antény

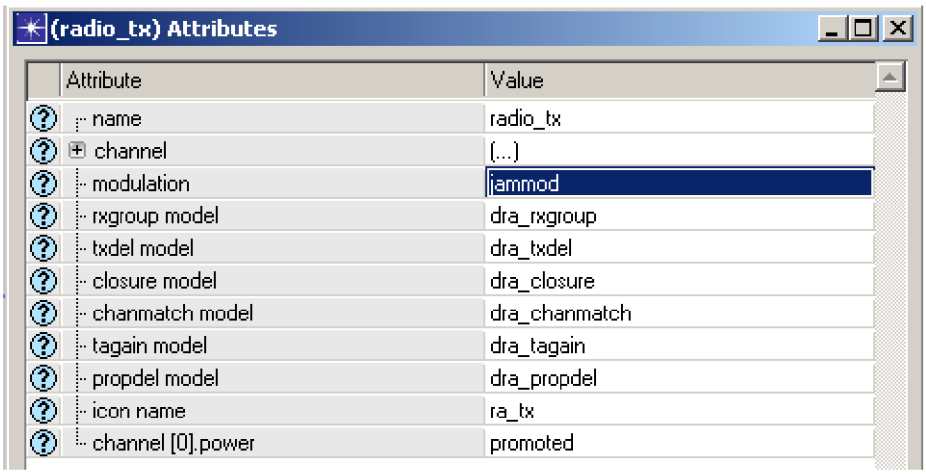
4.1.2 Rušící uzel

Rušící uzel (obr. 4.11) sítě generuje šum uvnitř sítě. Sestává ze stejných komponent jako stacionární vysílací uzel s tím rozdílem, že intenzita a modulace signálu jsou odlišné. Pakety s těmito rozdíly budou na přijímač působit jako šum.



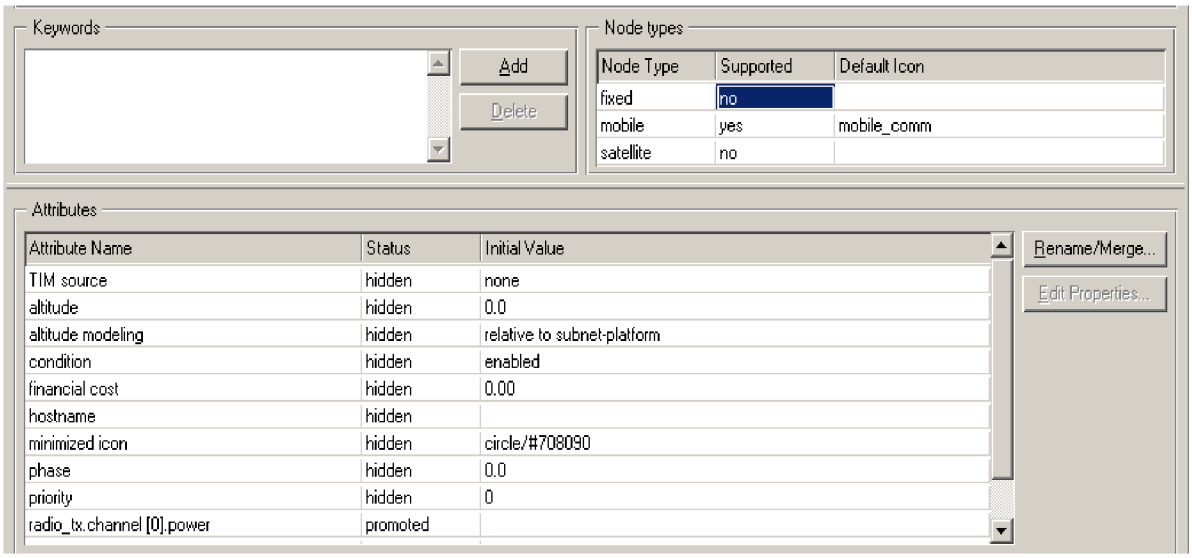
Obr. 4.11 Rušící uzel

Po kliknutí pravým tlačítkem myši na modulu *radio_tx* je vybrán *Edit Attributes* a poté nastaven parametr *modulation* na hodnotu *jammod* (obr. 4.12). Jde o typ modulace specifický pro rušící uzel s nulovou modulační křivkou, tedy závislostí BER na Eb/No, kde Eb/No vyjadřuje SNR na bit.



Obr. 4.12 Nastavení parametru *modulation* na hodnotu *jammod*

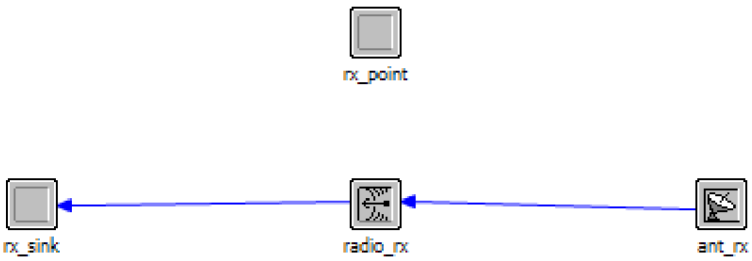
Také budou zadány atributy rozhraní uzlu podle obr. 4.13.



Obr. 4.13 Zadání atributů rozhraní rušícího uzlu

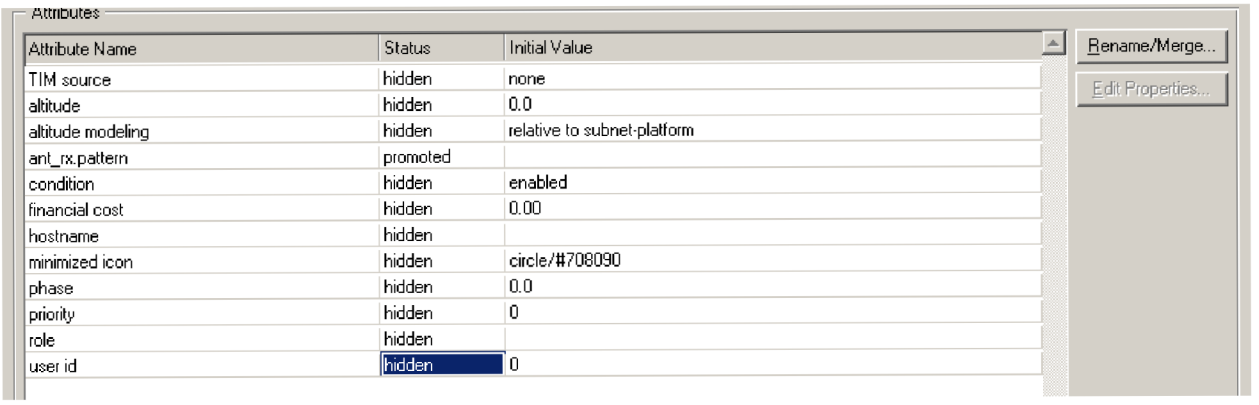
4.1.3 Přijímací uzel

Přijímací uzel (obr. 4.14) sestává z modulu antény, modulu přijímače, sink procesoru a zaměřovacího procesoru, který pomáhá nasměrovat anténu k vysílači.



Obr. 4.14 Přijímací uzel

Nyní budou zadány atributy rozhraní uzlu podle obr. 4.15.

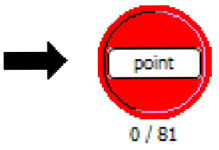


Obr. 4.15 Zadání atributů rozhraní uzlu přijímače

Zaměřovací procesor

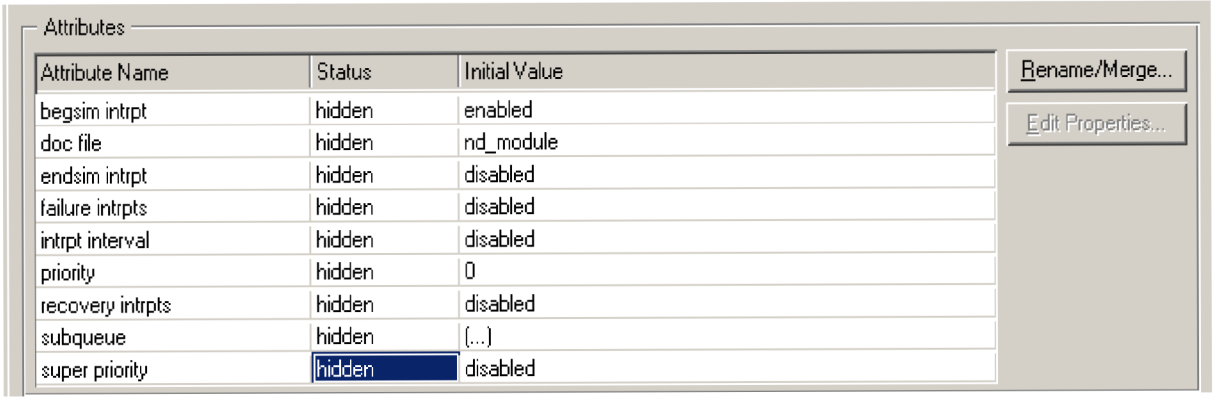
Zaměřovací procesor antény počítá polohu vysílače a nastavuje atributy zaměření modulu antény.

Pro modul rx_point bude vytvořen nový model procesu (obr. 4.16).



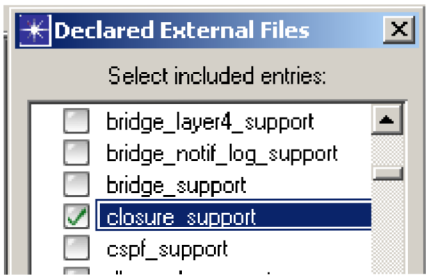
Obr. 4.16 Modul procesu

Do přidaného stavu je prostřednictvím okna vepsán kód Enter Executives viz příloha a. Nyní je potřeba upravit atributy procesu podle obr. 4.17.



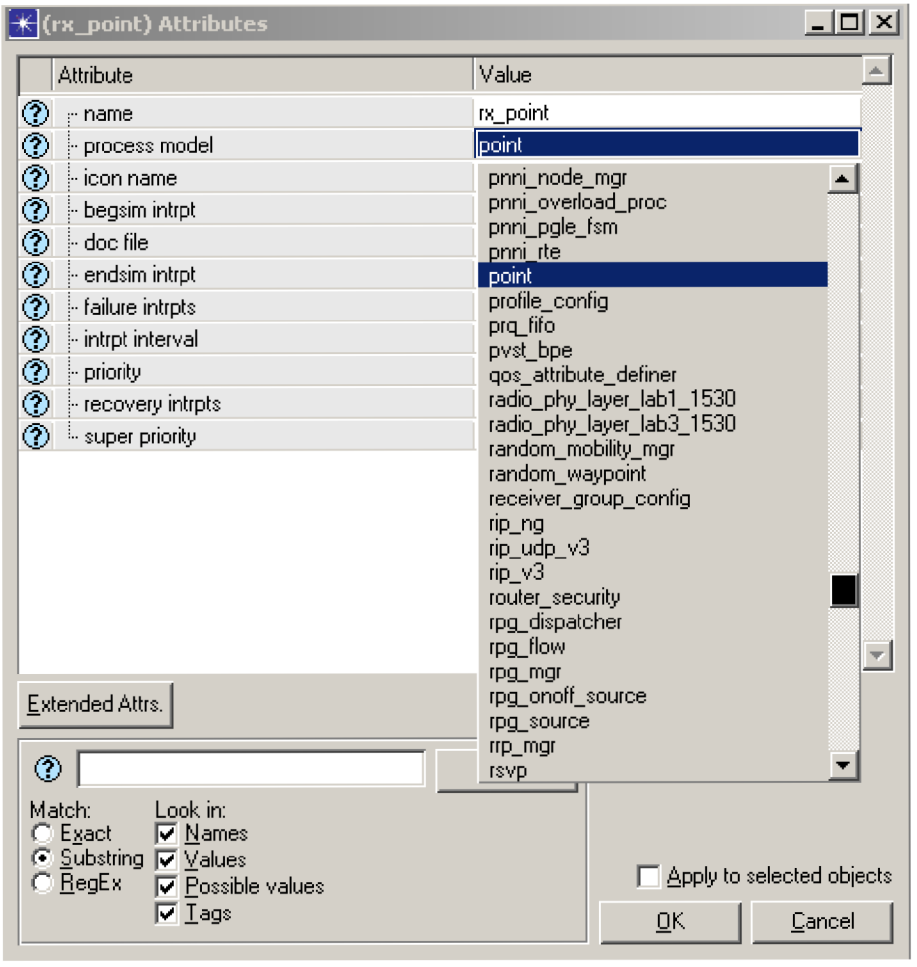
Obr. 4.17 Nastavení atributů procesu

Bezdrátová síť používá k přenosu paketů radiový přenos. Jelikož síť v této práci užívá předem definovaný model chování radiového přenosu (*dra_closure*), bude potřeba importovat knihovnu obsahující potřebné funkce. Následující procedura zajišťuje, aby byla tato knihovna při simulaci načtena (obr. 4.18).



Obr. 4.18 Nastaveni knihovny

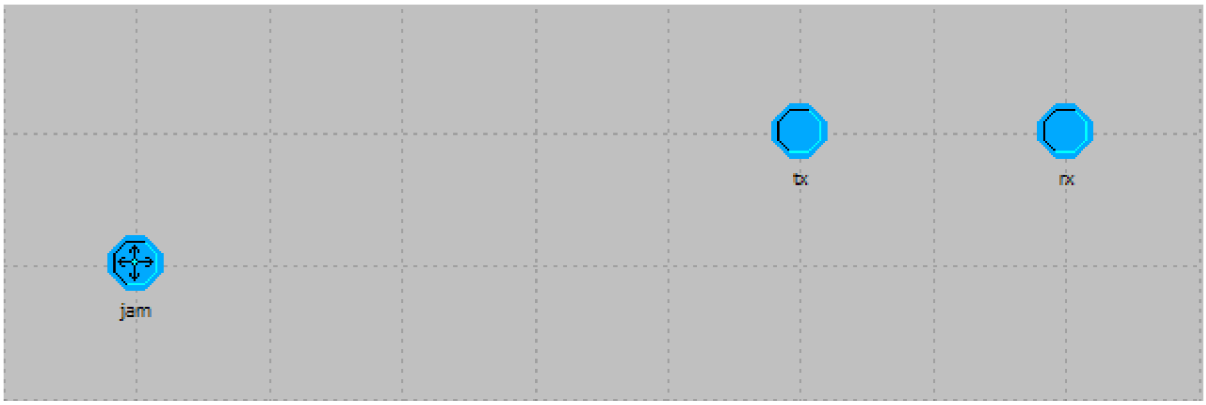
Celý model procesu bude uložen a přidán do modulu *rx_point* (obr. 4.19)



Obr. 4.19 Přidaní procesu do modulu *rx_point*

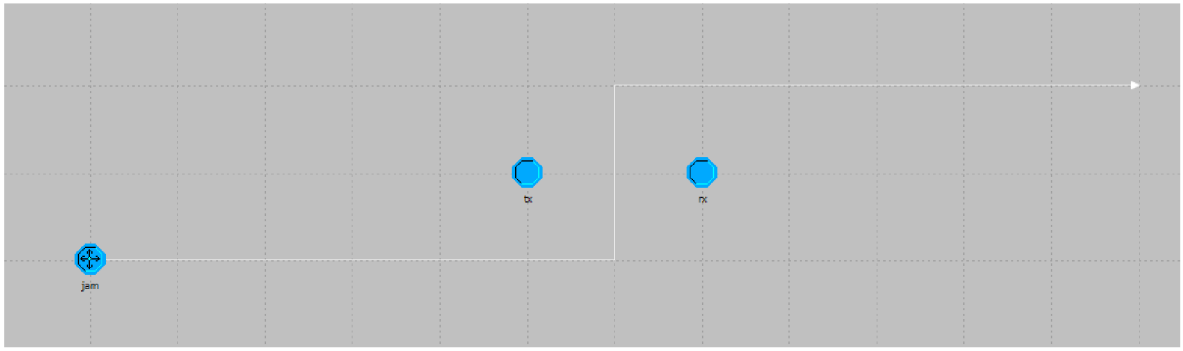
4.1.4 Vytváření modelu sítě

Je zadán název projektu a název scénářů. Následuje vytvoření prázdného scénáře výběrem *Create empty scenario* s použitím rozlohy sítě *Campus*, dále jsou vytvořeny všechny potřebné uzly. Vytvořený model sítě je na obr. 4.20.



Obr. 4.20 Model sítě

Když je vytvořen model sítě, je třeba definovat trajektorii rušícího uzlu podle obr. 4.21. Tím lze ověřit vliv rušení na kvalitu přenosu.



Obr. 4.21 Trajektorie rušícího uzlu

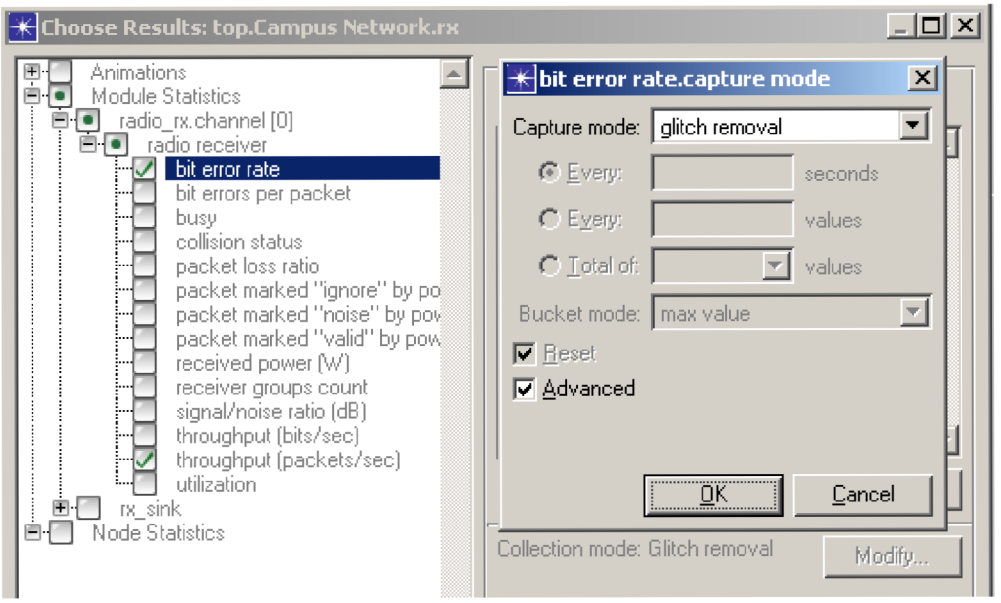
4.2 Nastavení záznamu statistik simulace

4.2.1 Záznam statistických údajů a běh simulace

Statistiky přijímaného kanálu mohou být zaznamenávány v editoru projektu. Tyto statistiky zahrnují BER a propustnost kanálu v paket/s. Propustnost kanálu je definována jako průměr správně přijatých paketů za sekundu. Nové záznamy v této statistice jsou generovány pouze pro pakety s nižší BER, než je hranice přijímače ECC zadané v modulu přijímače atributem *ecc threshold*.

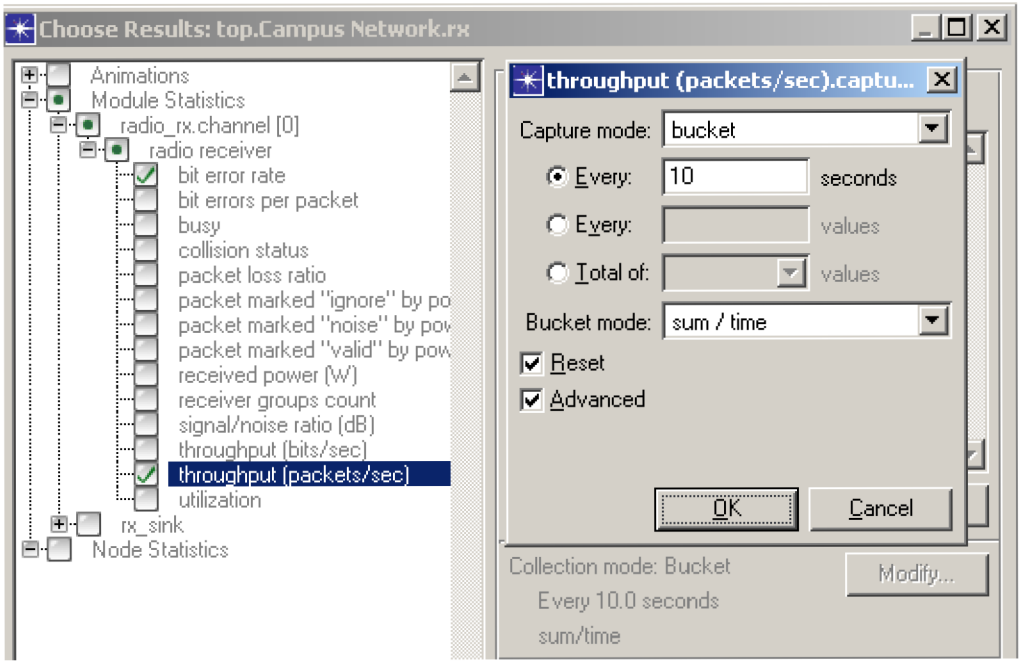
Jelikož modul radiopřijímače v této úloze má hodnotu tohoto atributu 0.0, budou přijaty pouze pakety s nulovou bitovou chybovostí.

Pro záznam statistiky BER a propustnost kanálu budou provedena následující nastavení pro modul rx (obr. 4.22).



Obr. 4.22 Nastavení parametru BER

Pro nastavení záznamového módu pro statistiku propustnosti kanálu budou provedena následující nastavení (obr. 4.23).



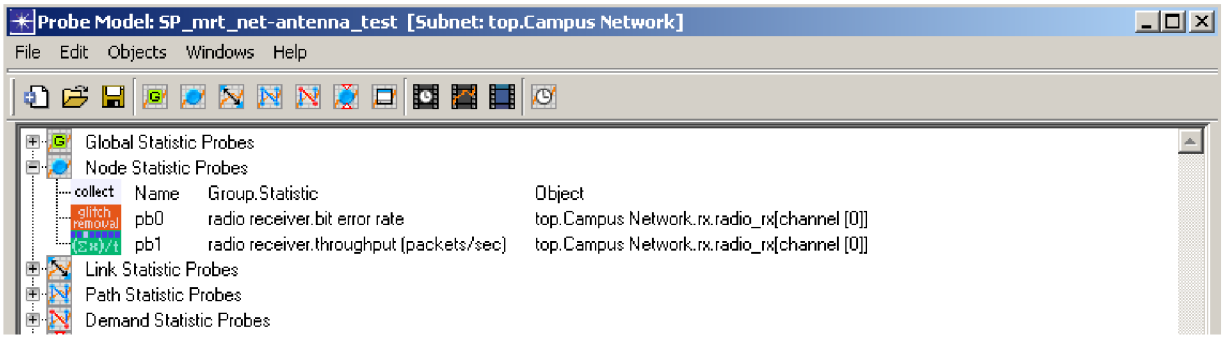
Obr. 4.23 Nastavení statistiky propustností kanálu

4.2.2 Záznam výsledků s pomocí editoru stanovení parametrů statistik.

V této části práce bude:

- prezentován editor stanovení parametrů statistik,
- předvedeny různé statistiky včetně několika takových, které nejsou přístupné v editoru projektu,
- konfigurována různá nastavení pro sběr statistických dat.

Z editoru projektu jsou již nastaveny dvě statistiky: BER a propustnost kanálu. Výběr jednotlivé DES statistiky z menu pravého tlačítka myši spustí jednoduché grafické rozhraní pro základní záznamy definovaných statistik (obr. 4.24).

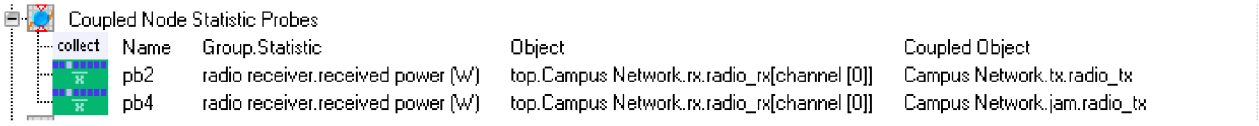


Obr. 4.24 Nastavení parametrů statistiky

Editor stanovení parametrů statistik bude zapisovat všechny hodnoty do specifikovaných záznamů. V případě bezdrátové komunikace je i dále možné podle původu dat omezit jejich počet. K tomu bude použit společný výběr statistik. Tento výběr je definován jak pro nastavení sběru statistických dat, tak pro příslušný vysílací a přijímací uzel. Zaznamenána budou pouze data plynoucí z komunikace mezi dvěma uzly.

V této části budou zaznamenávány jednotlivé příspěvky vysílání z rušícího a vysílacího uzlu na přijímaný výkon.

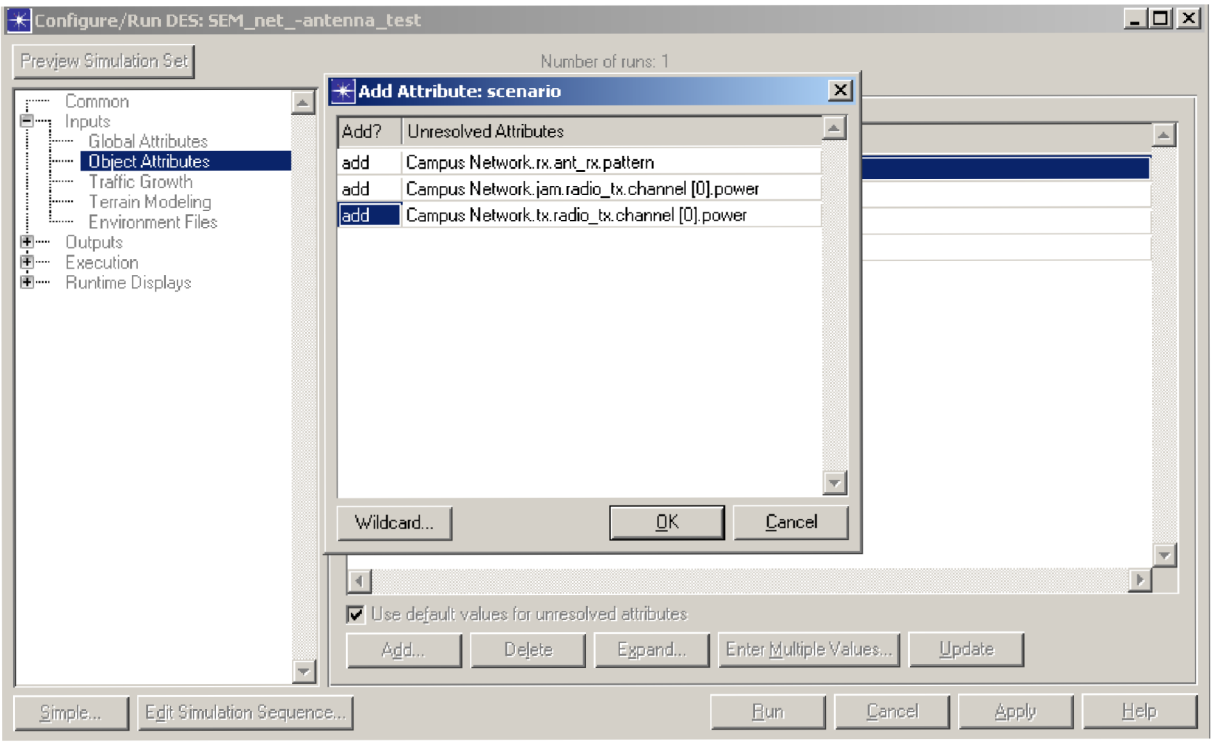
Nejdříve je nastavena společná volba statistik mezi vysílačem a přijímačem, jak je ukázáno na obr. 4.25.



Obr. 4.25 Nastaveni statistik mezi vysílačem a přijímačem

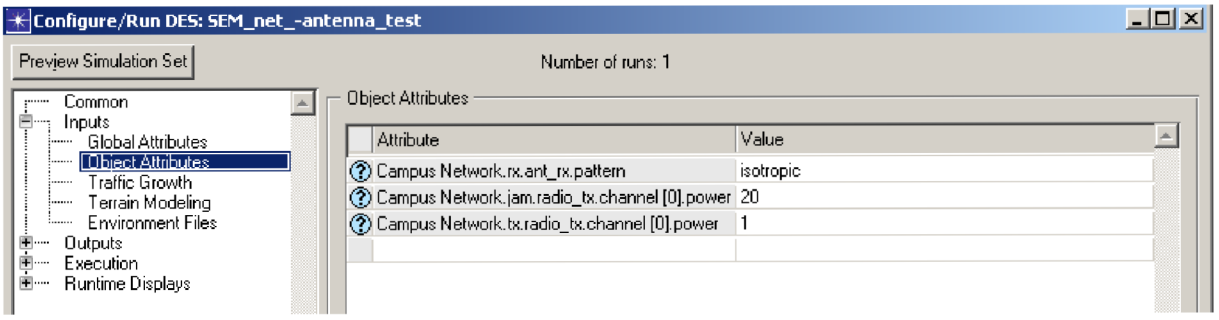
4.2.3 Nastavení a běh simulace

Nyní, když je nastaven záznam statistických dat (obr. 4.26), je možné konfigurovat simulaci prostřednictvím atributů, jejichž různé hodnoty mají rozmanitý vliv na chování sítě.



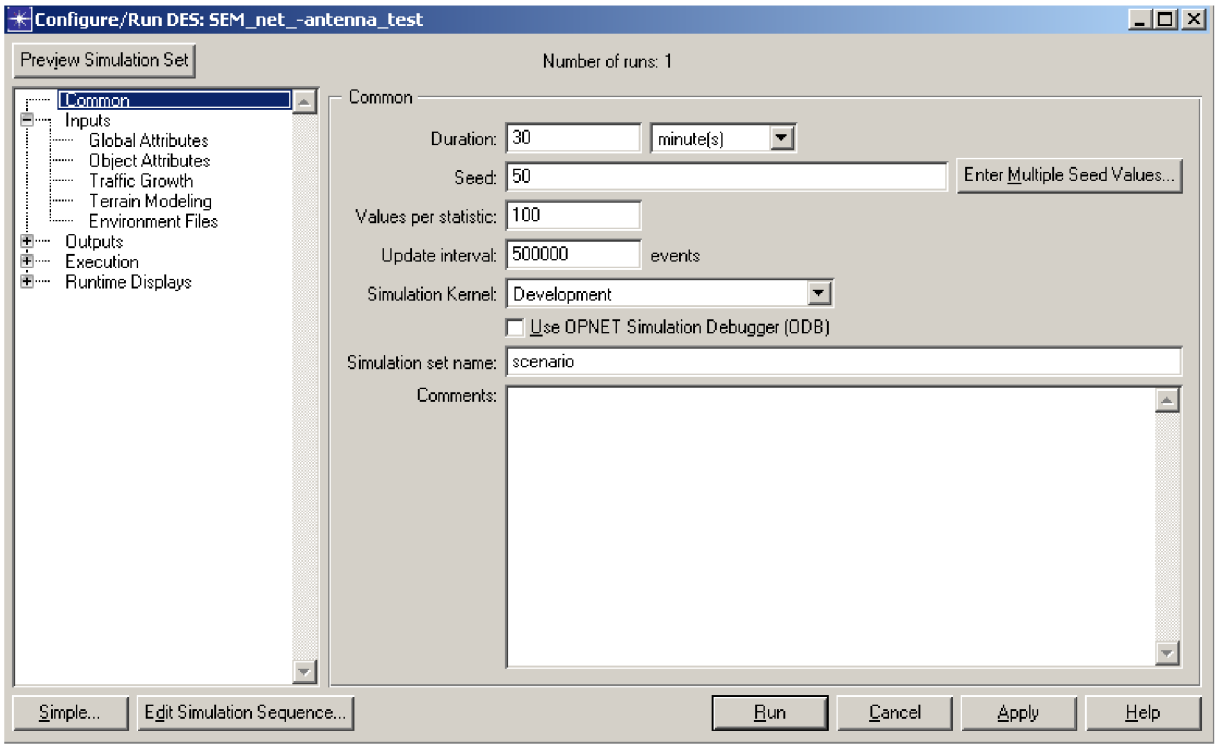
Obr. 4.26 Nastavení záznamu statistických dat

Těmto atributům je třeba ještě přidělit hodnoty, jelikož jim při výběru přiřazeny nebyly. Zadání atributů typu antény (*ant_rx.pattern*), výkon rušícího uzlu (*jam.radio_tx.channel[0].power*) a výkon vysílače (*tx.radio_tx.channel[0].power*) ukazuje obr. 4.27.



Obr. 4.27 Zadání atributů antény a výkonu vysílacího a rušícího uzlu

Změnu začátku a doby simulace popisuje následující nastavení (obr. 4.28):



Obr. 4.28 Nastavení doby simulace

4.3 Přehled a zpracování výsledků

Jakmile je simulace spuštěna, je možné ověřit chování sítě a zkontrolovat BER a propustnost kanálu.

4.3.1 Tabulkové statistiky

Pro lepší sledování chování sítě s různými typy antény je možné prohlédnout záznam celkové statistiky paketů pro každou běžící simulaci. Tyto záznamy obsahují statistiky vytvořených, kopírovaných a odstraněných paketů, poškození uzlů, moduly a formát paketů. Soupis se nachází v prohlížeči výsledků pod položkou **DES Run Tables. Run** obsahuje výsledky pro izotropní anténu. Je zřejmé, že během simulace bylo vytvořeno 1418 paketů. Vysílač a rušící uzel generovali pakety během simulace v časovém intervalu 10s – 12 min. Rychlost generování paketů byla 1 paket za sekundu, celkový počet paketů se tedy rovná *(12 x 60) - 10 = 710* paketů (tab. 4.1).

Tab. 4.1 Počet generovaných paketů během simulace

Název uzlu	Počet paketů
Zdroj rušení (campus network.jam)	710
Vysílač (campus network.tx)	710
Přijímač (campus network.rx)	
Celkový počet paketů	**1420**

Tab. 4.2 ukazuje, že přijímací uzel odstranil během simulace 1418 paketů Všechny kromě posledních paketů vysílače a rušícího uzlu, které byly v době ukončení simulace stále na cestě.

Tab. 4.2 Počet odstraněných paketů

Název uzlu	Počet paketů
Zdroj rušení (campus network.jam)	
Vysílač (campus network.tx)	
Přijímač (campus network.rx)	1418
Celkový počet paketů	**1418**

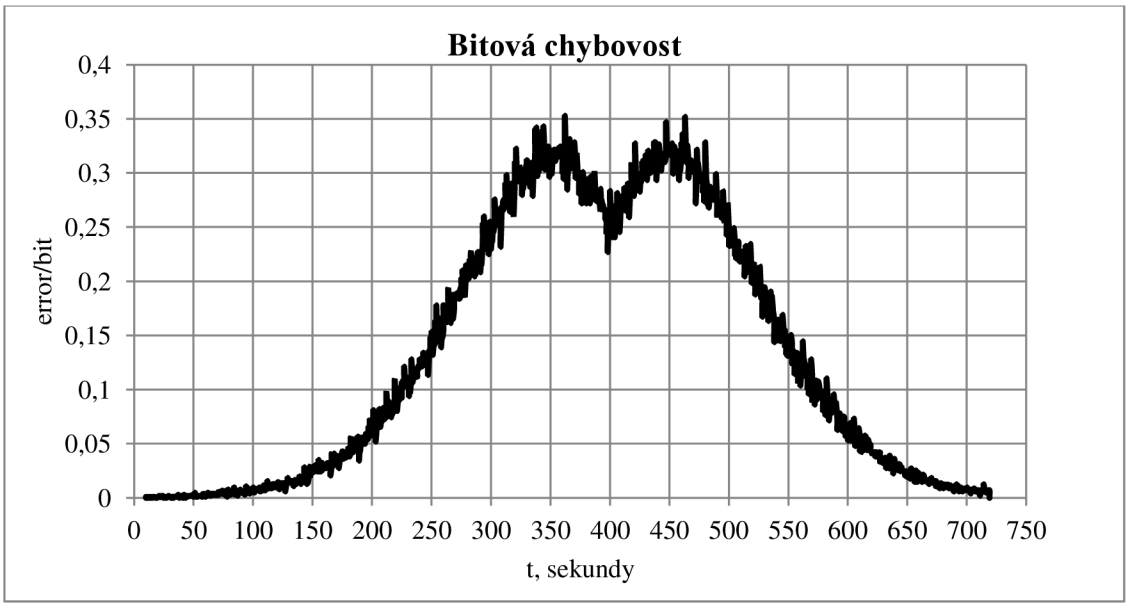
Modul *radio_rx* odstraňuje pakety, které byly narušeny interferencí rušícího uzlu. Správně přijaté pakety jsou odstraněny v modulu *rx_sink* (tab. 4.3).

Tab. 4.3 Počet odstraněných paketů v modulu radio_rx

Název modulu	Počet paketů
Anténa prijínače (ant_rx)	
Přijímač (radio_rx)	1405
Zaměřovací procesor antény (rx_point)	
Sink procesor (rx_sink)	13
Celkový počet paketů	**1418**

4.3.2 Grafické statistiky

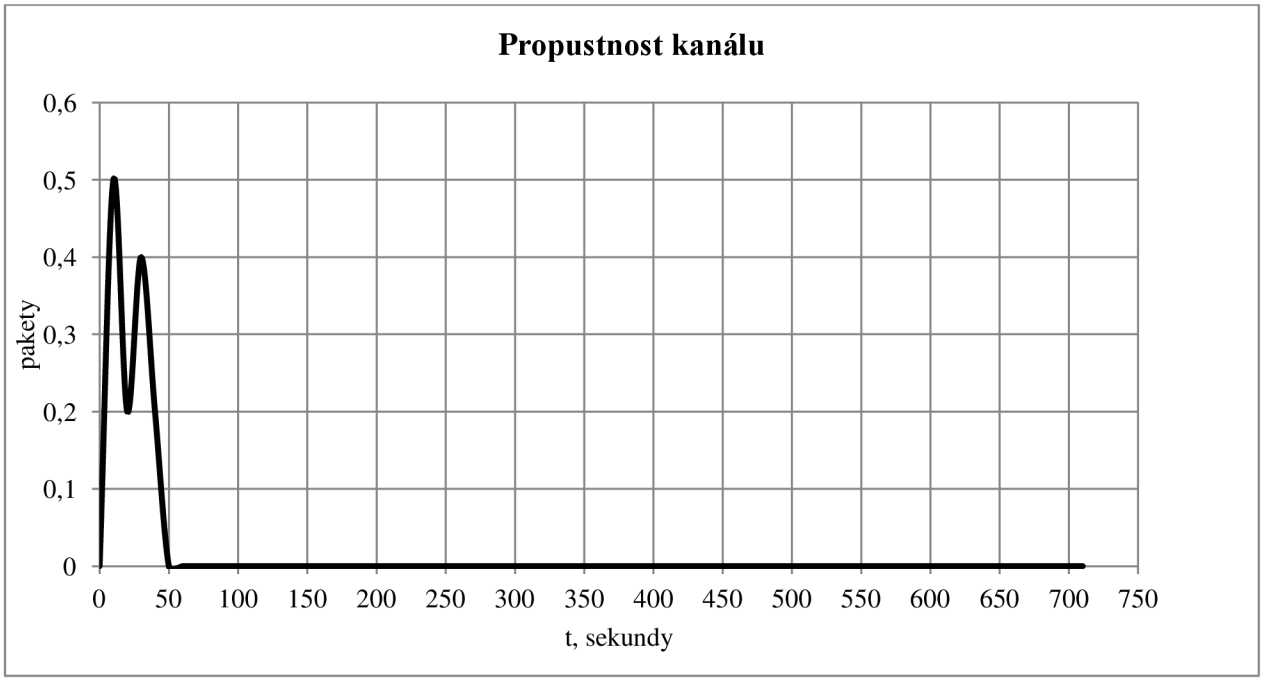
Nyní, když je znám vliv různých antén na statistiky příjmu paketů, bude ověřen BER (obr. 4.29) a propustnost kanálu (obr. 4.30).



Obr. 4.29 Grafické statistiky: BER

Podle předpokladů, grafické statistky pro izotropní anténu ukazují, jak na přijímači hodnota BER postupně narůstá se zmenšující se vzdáleností mezi rušícím a přijímacím uzlem a naopak.

BER dosáhl maxima okolo 0,35 chyb/bit při nejmenší vzdálenosti mezi rušícím a přijímacím uzlem. Izotropická přijímací anténa přijímala interferenci rušícího uzlu během celé simulace. Dvě maxima BER jsou dosažena při maximálním přiblížení rušícího uzlu od přijímače.



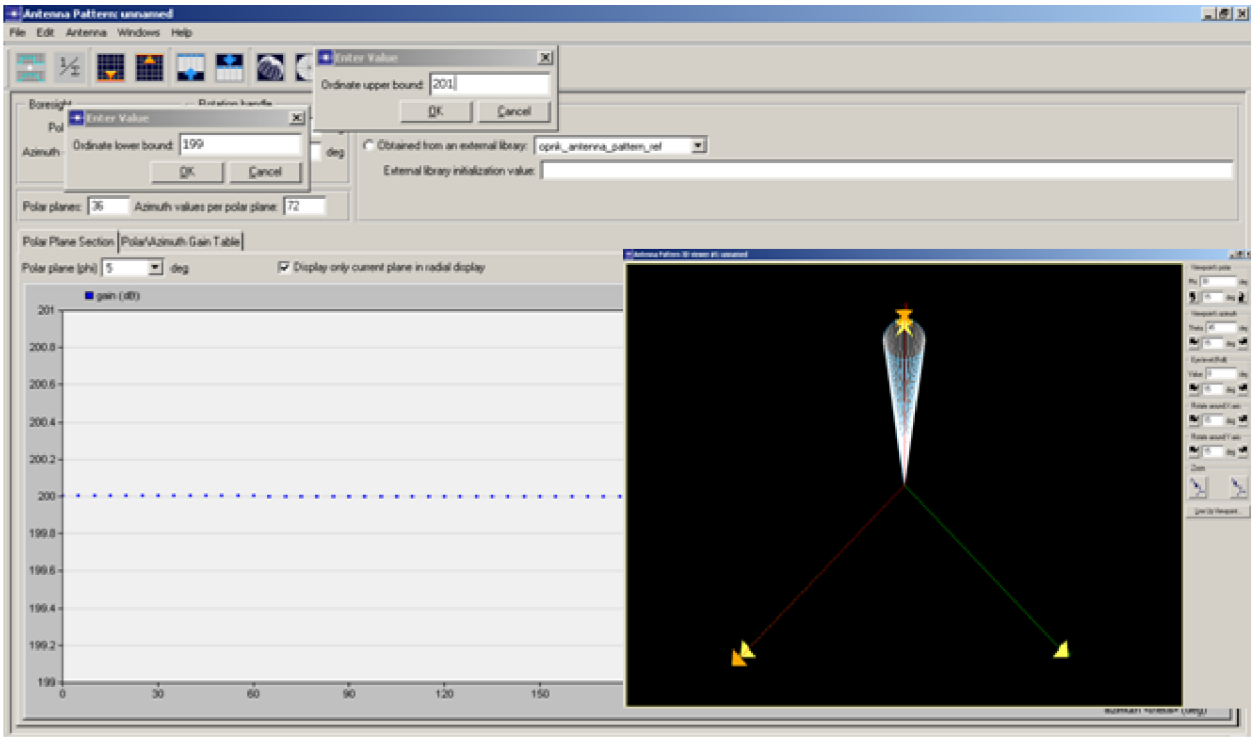
Obr. 4.30 Grafické statistiky: propustnost kanálu

5 SÉRIE SIMULACÍ

Tato část práce vychází z předchozí části a je v ní provedena série simulací s pozměněnými parametry. Cílem těchto simulací je ověřit vliv těchto parametrů na kvalitu přenosu.

5.1 Změna typu antény a typu modulace

Anténa používá dva odlišné vzory, izotropní, který má stejný zisk ve všech směrech, a směrový, který bude definován (obr. 5.1).



Obr. 5.1 Směrová anténa

Základní model bezdrátové sítě obsahuje rušící uzel, který se pohybuje kolem přijímače a vysílače podle trajektorie (obr. 4.20). Pro všesměrovou a směrovou anténu byly použity typy modulace BPSK (Binary-Phase Shift Keying) (výchozí nastavení), PSK8 (Phase Shift Keying) a QAM64 (Quadrature Amplitude Modulation) (tab 5.1, obr. 5.2 a obr. 5.3).

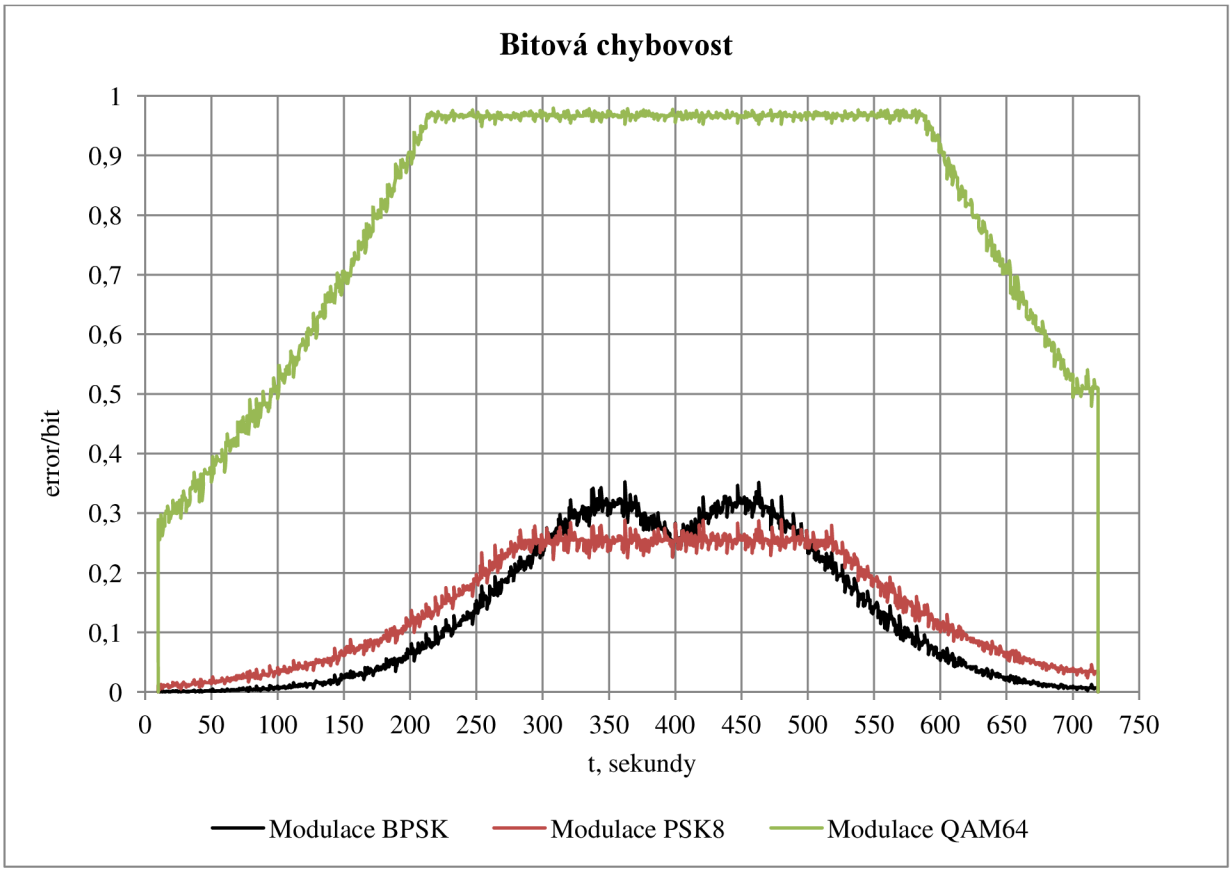
BPSK je dvoustavová modulace založená na posunutí fáze nosné frekvence na 0 nebo 180 stupnů.

PSK8 je druh vícestavové fázové modulace, kde se předpokládá, že fáze nosného signálu může nabývat osmi různých hodnot. Modulace PSK8 umožňuje v jednom symbolu přenést 3 bity 2^3=8 modulačních stavů.

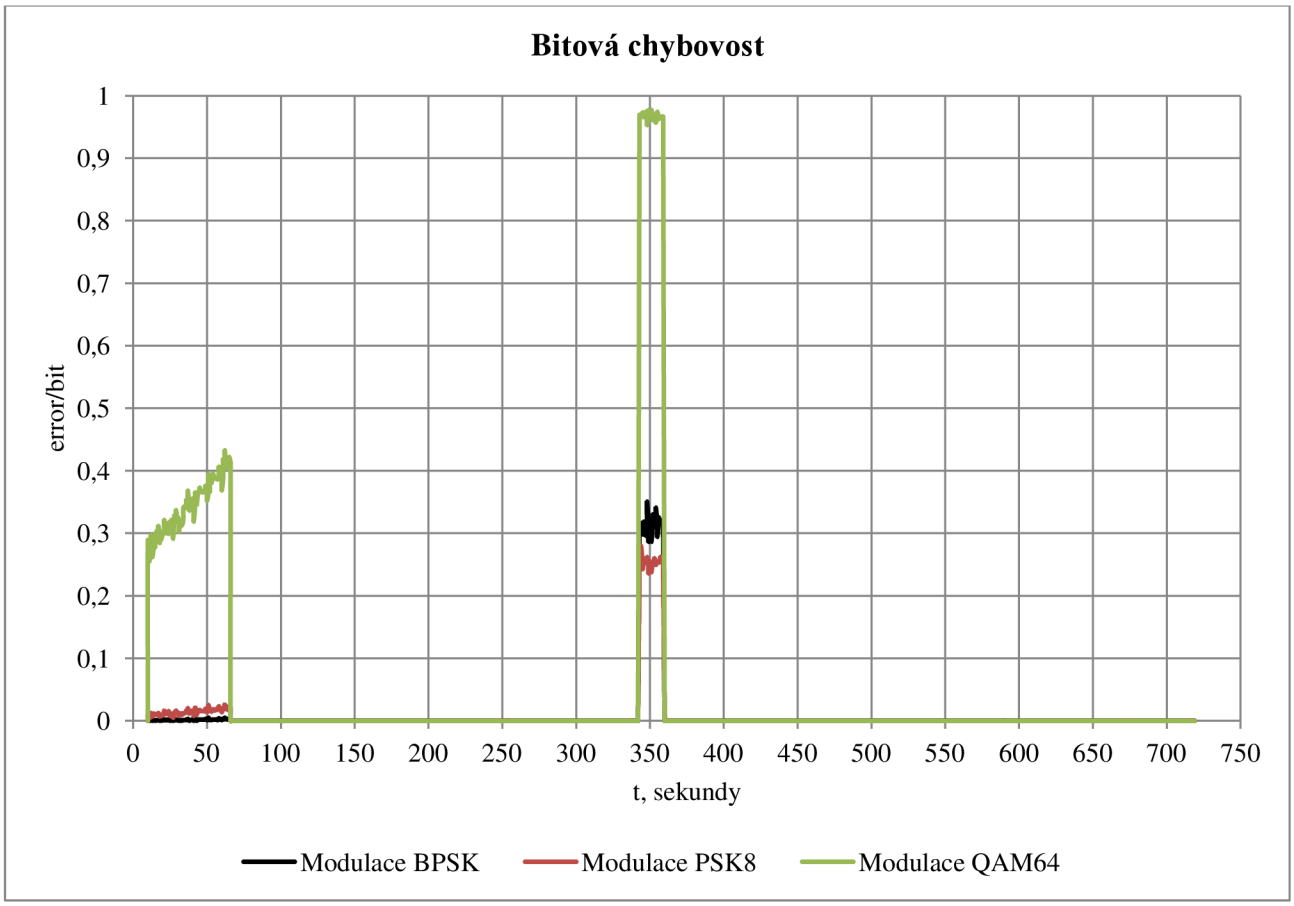
QAM64 představuje vícestavovou modulaci, která k vytváření symbolů využívá kombinaci amplitudového a fázového klíčování. Každý stav je reprezentován určitou hodnotou amplitudy a fáze. Modulace QAM64 umožňuje v jednom symbolu přenést 6 bitů 2^6=64 modulačních stavů.

Tab. 5.1 Výsledky příjmu paketů pro různé typy modulace

Uzel modelu	BPSK		PSK8		QAM64	
	všesměrová	směrová	všesměrová	směrová	všesměrová	směrová
ant_rx [paket]	0	0	0	0	0	0
radio_rx [paket]	1405	769	1418	783	1418	782
rx_point [paket]	0	0	0	0	0	0
rx_sink [paket]	13	649	0	635	0	636
Úspěšnost [%]	1,83	91,54	0	89,56	0	89,7



Obr. 5.2 BER pro různé typy modulace při použití všesměrové antény

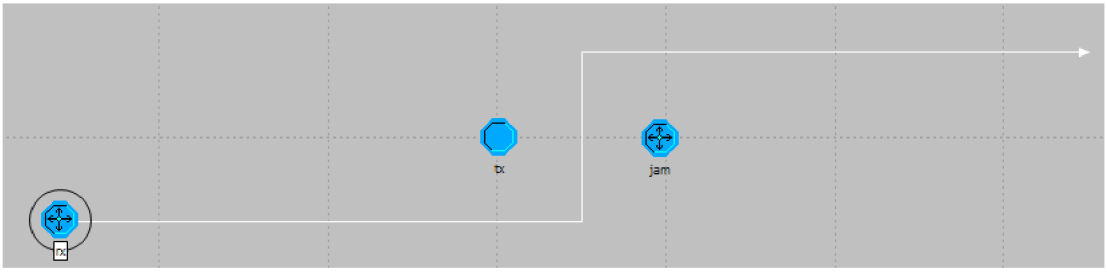


Obr. 5.3 BER pro různé typy modulace při použití směrové antény

Z obr. 5.2 a obr. 5.3 je patrné, že přijímač se všesměrovou anténou je ovlivňován rušícím uzlem ze všech směrů, zatímco přijímač se směrovou anténou je ovlivňován rušícím uzlem pouze v případě, když se nachází v jejím směru. Vliv antény lze vyčíst z tab. 5.2.

5.2 Změna topologie sítě

V dalším scénáři byly provedeny změny topologie oproti základnímu modelu. V následujících modelech se pohybuje přijímač kolem vysílacího a rušícího uzlu (obr. 5.4, obr. 5.6 a obr. 5.8). Počet ztracených paketů závisí na vzdálenosti rušícího od přijímacího uzlu.

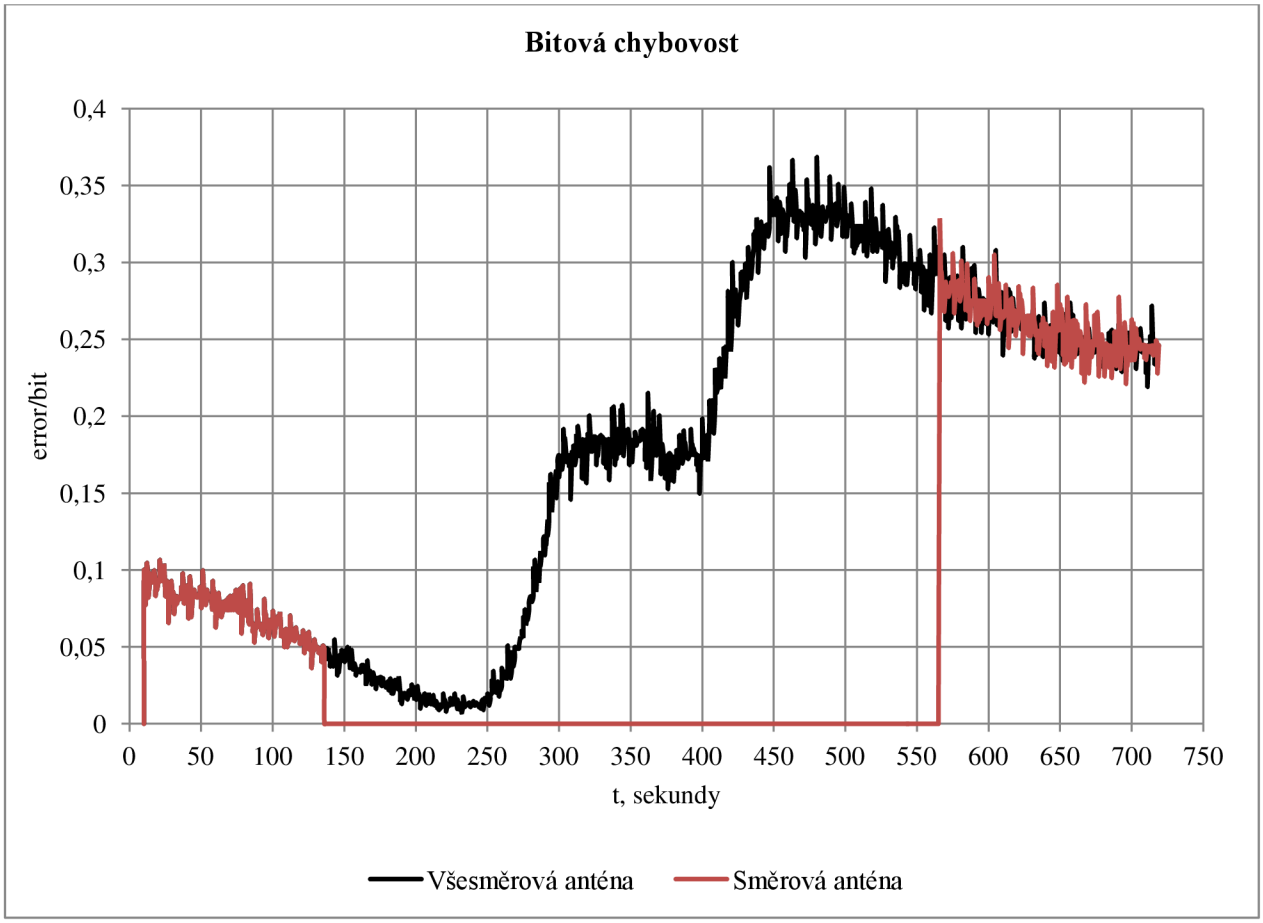


Obr. 5.4 Topologie 1

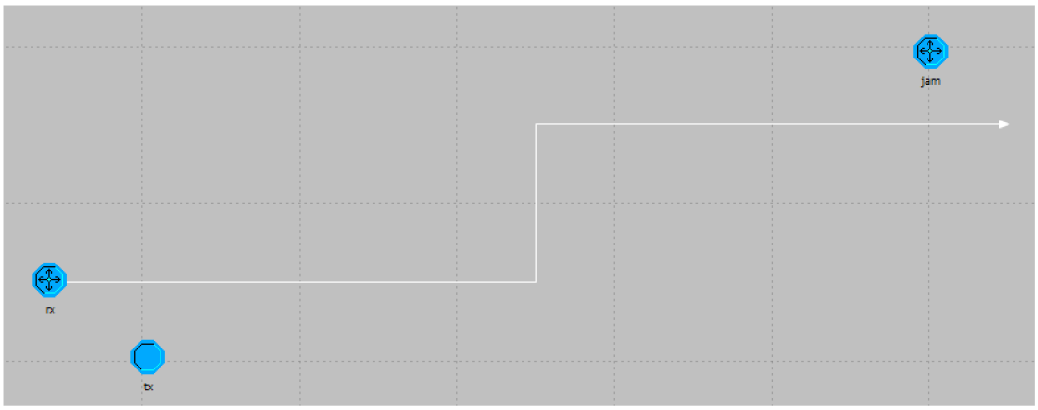
V této topologii jsou vysílací a rušící uzel stacionárními uzly a přijímací uzel se pohybuje podle trajektorie na obr. 5.4. Situace je obdobná jako u modelového příkladu, avšak rušící a přijímací uzel jsou zde zaměněny.

V případě všesměrové antény dosahuje BER minimální hodnoty 0,01 v čase 250 s. V intervalu času 300 s – 400 s je přijímací uzel vzdálen stejně od vysílacího a rušícího uzlu a BER má průměrnou hodnotu 0,18. V čase 450 s dosahuje BER maximální hodnoty 0,33.

V případě směrové antény se v intervalech (10 – 135) s a (565 – 720) s rušící uzel nachází ve směrovém úhlu přijímací antény a hodnota BER kopíruje výsledky získané s použitím všesměrové antény. V intervalu 235 – 565 s přijímací uzel nepřijímá rušení rušícího uzlu a hodnota BER je nulová.



Obr. 5.5 BER pro topologii 1

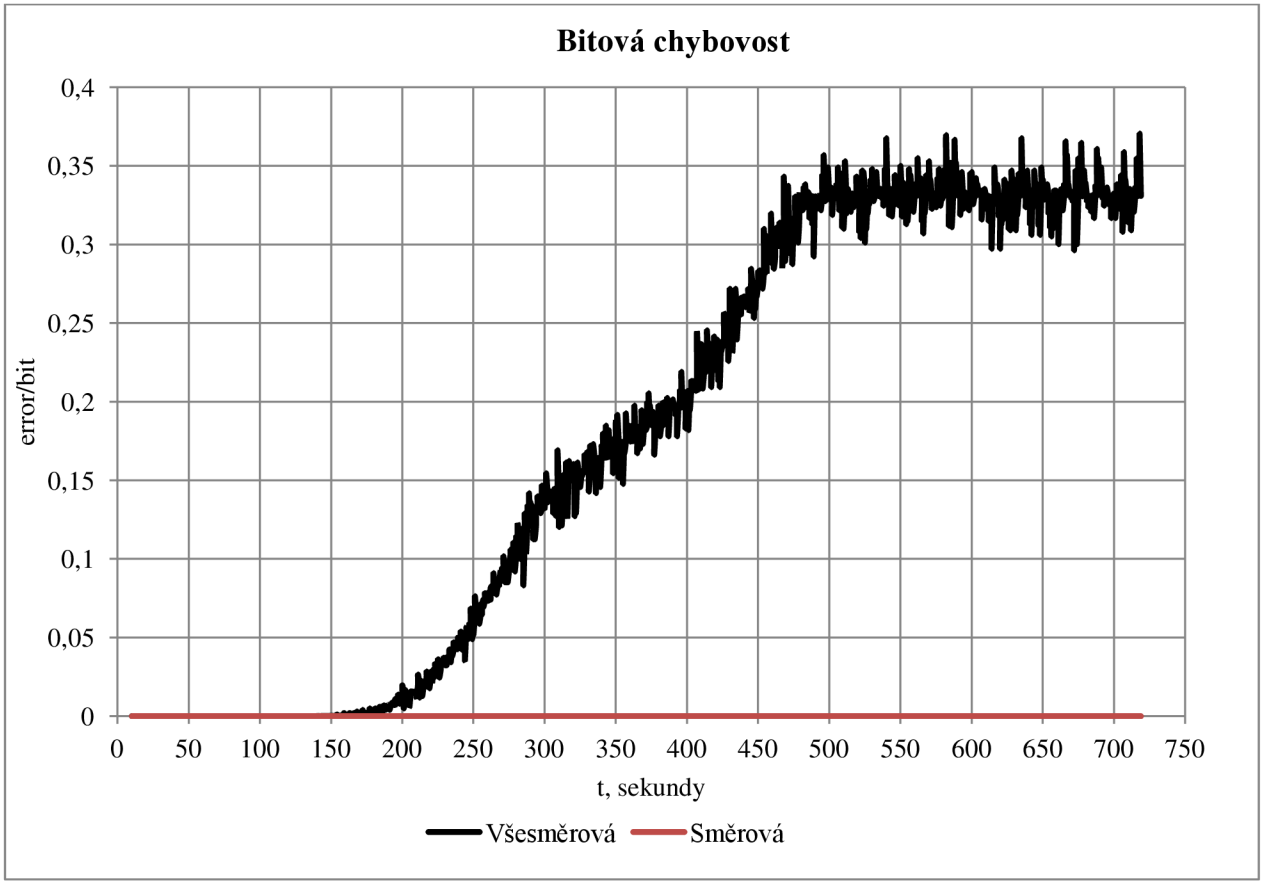


Obr. 5.6 Topologie 2

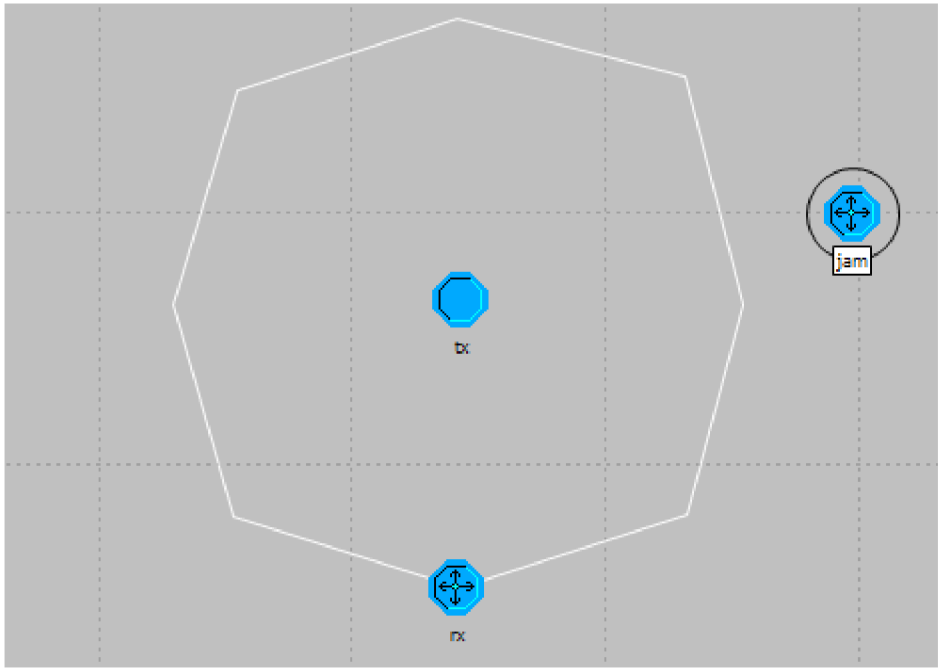
V topologii č. 2 jsou opět vysílací a rušící uzel stacionární a přijímací uzel se pohybuje podle trajektorie na obr. 5.6.

V případě všesměrové antény je v intervalu (10 – 150) s rušící uzel dostatečně vzdálen a hodnota BER je nulová. Od tohoto intervalu se bitová chybovost se zmenšující se vzdáleností od rušícího uzlu zvyšuje.

V případě směrové antény se během celé simulace rušící uzel nenachází ve směrovém úhlu přijímacího uzlu a hodnota BER je nulová.



Obr. 5.7 BER pro topologii 2

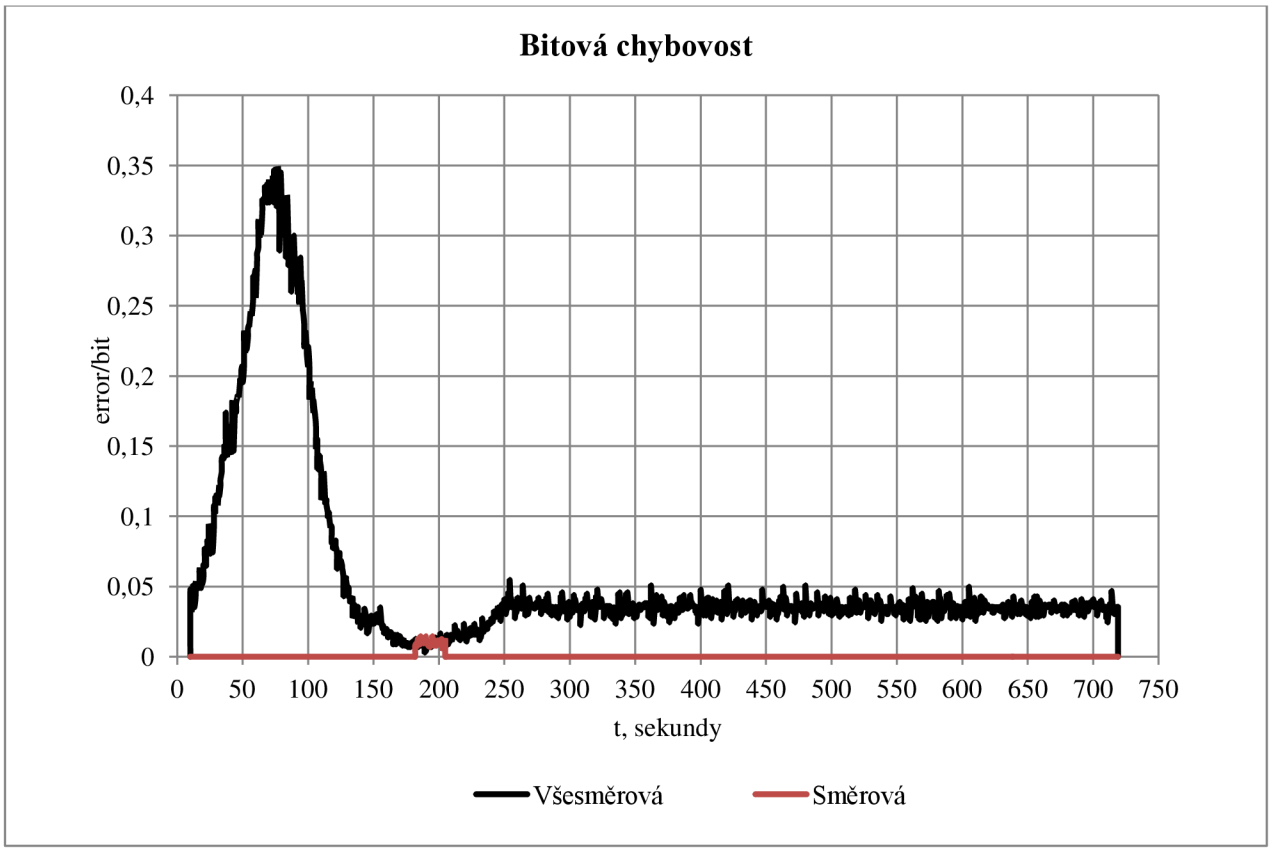


Obr. 5.8 Topologie 3

V této topologii č. 3 jsou opět vysílací a rušící uzel stacionárními uzly a přijímací uzel se pohybuje podle trajektorie na obr. 5.8. Trajektorie je kruhová se směrem pohybu proti směru hodinových ručiček. Simulován byl jeden oběh s periodou 250 s.

V případě všesměrové antény se v čase 75 s přijímací uzel nachází na přímce mezi rušícím a vysílacím uzlem a hodnota BER dosahuje maxima 0,35. V čase 175 s se přijímací uzel nachází opět na přímce v nejvyšší vzdálenosti od rušícího uzlu a hodnota BER je naopak nejmenší s hodnotou 0,01.

V případě směrové antény se v čase 175s rušící uzel nachází ve směrovém úhlu přijímací antény a v okolí tohoto bodu dochází ke zvýšení BER z nuly na hodnotu 0,01.



Obr. 5.9 BER pro topologii 3

Tab. 5.2 Výsledky příjmu paketů pro různé typy topologie

Uzel modelu	**Topologie 1**		**Topologie 2**		**Topologie 3**	
	všesměrová	směrová	všesměrová	směrová	všesměrová	směrová
ant_rx [paket]	0	0	0	0	0	0
radio_rx [paket]	1418	988	1264	709	1418	732
rx_point [paket]	0	0	0	0	0	0
rx_sink[paket]	0	430	154	709	0	686
Úspěšnost [%]	0	60,65	21,72	100	0	97,18

6 POPIS TDA RÁDIOVÉHO PŘENOSU

TDA je proměnná libovolného datového typu představující atribut paketu datového přenosu. TDA umožňuje přenos informací mezi simulačním jádrem a externě definovanými procedurami, které popisují model přenosu. Tyto procedury tvoří kroky přenosu popsané v kapitole 3.

Simulační jádro vyčleňuje základní sadu TDA modelům přenosu volaným jednotlivými kroky přenosu během simulace a podporuje komunikaci mezi jádrem a kroky přenosu stejně jako mezi jednotlivými kroky. Sada TDA vyhrazená pro radiový přenos je definována v následující tabulce. Pro přehlednost jsou symbolické konstanty představující index TDA uvedeny ve zkrácené podobě bez jejich předpony. Skutečný název sestává z názvu v tab. 6.1 a předpony *OPC_TDA_RA_* (např. *OPC_TDA_RA_BER*).[7]

Tab. 6.1 TDA rádiového přenosu

Proměnná	Datový typ	Definice	Přidělení	Nastavitelné modelem kroku
ACTUAL_BER	double	podíl chybných bitů v paketu nebo jeho segmentů	12. krok	ano
BER	double	předpokládaná bitová chybovost paketu nebo jeho segmentů	11. krok	ano
BKGNOISE	double	šum pozadí vlivem nemodelových zdrojů šumu (ve Wattech)	8. krok	ano
CLOSURE	integer	schopnost vysílacího kanálu navázat spojení s přijímacím kanálem	2. krok	ano
ECC_THRESH	double	maximální počet chybných bitů, které přijímač může opravit	jádro	ano
END_DIST	double	vzdálenost mezi vysílačem a přijímačem v čase ukončení přenosu (v metrech)	jádro	ano
END_PROPDEL	double	doba šíření rádiového signálu mezi vysílačem a přijímačem v čase ukončení přenosu (v sekundách)	5. krok	ano
END_RX	double	čas ukončení příjmu paketu (čas zahájení přenosu + zpoždění ukončení příjmu)	jádro	ne
END_TX	double	čas ukončení přenosu paketu	jádro	ne
MATCH_STATUS	integer	vyhodnocení kompatibility mezi vysílacím a přijímacím kanálem (platný, šum, ignorovaný)	3. krok	ano

Proměnná	Datový typ	Definice	Přidělení	Nastavitelné modelem kroku
ND_FAIL	double	čas prvního selhání během přijmu paketu	jádro (pokud dojde k selhání)	ne
ND_RECOV	double	čas posledního oživení během přijmu paketu	jádro (pokud dojde k oživení)	ne
NOISE_ACCUM	double	seskupený interferenční šum v paketu nebo jeho segmentu (ve Wattech)	8. krok	ano
NUM_COLLS	integer	počet kolizí	resetování jádra	ano
NUM_ERRORS	integer	počet bitových chyb mající vliv na pakety	12. krok a resetování jádra	ano
PK_ACCEPT	integer	rozhodnuti o přijetí/odmítnutí paketu	13. krok	ano
PROC_GAIN	double	zisk přijímacího systému (v dB), používá se k výpočtu efektivního SNR	jádro	ano
RCVD_POWER	double	výkon přijímaného rádiového signálu (ve Wattech)	7. krok	ano
RX_BORESIGHT_PHI, RX_BORESIGHT_THETA	double	referenční bod přijímací antény (ve stupních), obvykle bod maximálního zisku	jádro	ano
RX_BW	double	šířka pásma přijímacího kanálu (v Hz)	jádro	ano
RX_CH_INDEX	integer	index přijímacího kanálu	jádro	ne
RX_CH_OBJID	integer	ID přijímacího kanálu	jádro	ne
RX_CODE	double	identifikační kód přijímacího kanálu	jádro	ano
RX_DRATE	double	přenosová rychlost přijímacího kanálu (v bps)	jádro	ano
RX_FREQ	double	nosný kmitočet přijímacího kanálu (v Hz)	jádro	ano
RX_GAIN	double	zisk antény přijímače	6. krok	ano
RX_GEO_X, RX_GEO_Y, RX_GEO_Z	double	kartézské souřadnice přijímacího uzlu (v metrech)	jádro	ano

Proměnná	Datový typ	Definice	Přidělení	Nastavitelné modelem kroku
RX_LAT,	double	zeměpisná šířka a délka (ve stupních), nadmořská výška (v metrech) přijímacího uzlu	jádro	ano
RX_LONG,				
RX_ALT				
RX_MOD	integer	identifikátor formátu modulace vysílačů	jádro	ano
RX_NOISEFIG	double	šumový činitel přijímače	jádro	ano
RX_OBJID	integer	ID přijímacího uzlu	jádro	ne
RX_PATTERN	integer	identifikátor zisku přijímací antény prostřednictvím její struktury	jádro	ano
RX_PHI_POINT,	double	směrové úhly (ve stupních) určené na základě souřadnic přijímací antény	jádro	ano
RX_THETA_POINT				
RX_REL_X,	double	pozice přijímacího uzlu v souřadnicovém systému vlastní podsítě	jádro	ano
RX_REL_Y				
SNR	double	poměr signál šum paketu nebo jeho segmentu měřený na přijímači v dB)	10. krok	ano
SNR_CALC_TIME	double	čas posledního výpočtu SNR	10. krok	ano
START_DIST	double	vzdálenost mezi vysílačem a přijímačem na začátku přenosu (v metrech)	jádro	ano
START_PROPDEL	double	doba šíření rádiového signálu mezi vysílačem a přijímačem na začátku přenosu (v sekundách)	5. krok	ano
START_RX	double	začátek přijmu paketu	jádro	ne
START_TX	double	začátek vysílaní paketu	jádro	ne
TX_BORESIGHT_PHI,	double	referenční bod vysílací antény (ve stupních)	jádro	ano
TX_BORESIGHT_THETA				
TX_BW	double	šířka pásma vysílacího kanálu (v Hz)	jádro	ano
TX_CH_INDEX	integer	index vysílacího kanálu	jádro	ne
TX_CH_OBJID	integer	ID vysílacího kanálu	jádro	ne
TX_CODE	double	identifikační kód vysílacího kanálu	jádro	ano

Proměnná	Datový typ	Definice	Přidělení	Nastavitelné modelem kroku
TX_DELAY	double	zpoždění vysílaní paketu	1. krok	ano
TX_DRATE	double	přenosová rychlost vysílacího kanálu (v bps)	jádro	ano
TX_FREQ	double	nosný kmitočet vysílacího kanálu (v Hz)	jádro	ano
TX_GAIN	integer	zisk antény vysilače	4. krok	ano
TX_GEO_X,	double	kartézské souřadnice vysílacího uzlu (v metrech)	jádro	ano
TX_GEO_Y,				
TX_GEO_Z				
TX_LAT,	double	zeměpisná šířka a délka (ve stupních), nadmořská výška (v metrech) vysílacího uzlu	jádro	ano
TX_LONG,				
TX_ALT				
TX_MOD	integer	identifikátor formátu modulace vysilače	jádro	ano
TX_OBJID	integer	ID vysílacího uzlu	jádro	ne
TX_PATTERN	integer	identifikátor zisku vysílací antény prostřednictvím její struktury	jádro	ano
TX_PHI_POINT,	double	směrové úhly (ve stupních) určené na základě souřadnic vysílací antény	jádro	ano
TX_THETA_POINT				
TX_POWER	double	výkon vysílaní (ve Wattech)	jádro	ano
TX_REL_X,	double	pozice vysílacího uzlu v souřadnicovém systému vlastní podsítě	jádro	ano
TX_RAL_Y				

OM umožňuje dynamické modelování mezi komunikujícími uzly. Charakteristiky a dostupnost těchto uzlů mohou tedy záviset na mnoha faktorech. Každý ze 14. modelů reprezentujících kroky přenosu je popsán příslušným kódem v programovacím jazyce C++, který lze nadále modifikovat dle potřeb uživatele (tab. 6.2). [7]

Tab. 6.2 Výchozí kroky rádiového přenosu OM

Kroky	Název modelu v OM	Název výchozího kódu
0. Skupina přijímačů	rxgroup model	dra_rxgroup
		dra_rxgroup_no_rxstate
1. Zpoždění přenosu	txdel model	dra_txdel
2. Rezervace linky	closure model	dra_closure
3. Výběr kanálu	chanmatch model	dra_chanmatch
4. Zisk vysílací antény	tagain model	dra_tagain
5. Zpoždění vysílaní	propdel model	dra_propdel
6. Zisk přijímací antény	ragain model	dra_ragain
7. Síla přijatého signálu	power model	dra_power
		dra_power_no_rxstate
8. Interferenční šum	inoise model	dra_ inoise
9. Šum pozadí	bkgnoise model	dra_ bkgnoise
10. Poměr signál - šum	snr model	dra_snr
11. Bitová chybovost	ber model	dra_ber
12. Alokace chyb	error model	dra_error
13. Korekce chyb	ecc model	dra_ecc
		dra_ecc_no_rxstate

Prostřednictvím výše zmíněných atributů jednotlivých kroků přenosu lze během simulace číst jejich aktuální hodnoty a ty dále zobrazit v simulační konzoli OM.

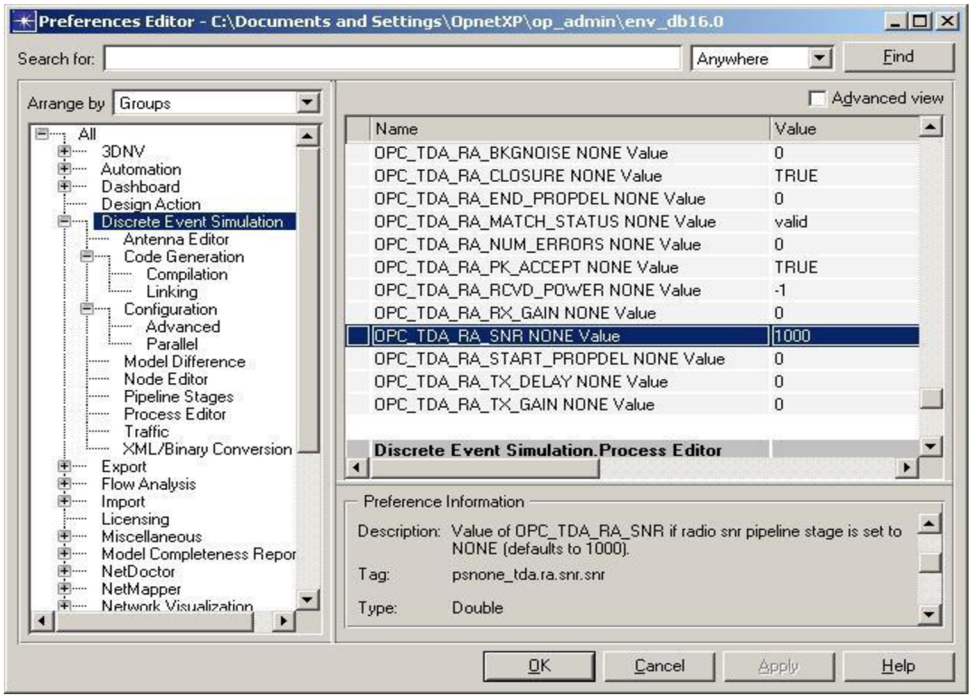
Jestliže některý z kroků přenosu dat není v simulaci důležitý, lze tyto kroky přeskočit. Pro přeskočení kroku je třeba do řádku atributu přiřadit hodnotu „NONE“. Simulační jádro krok přeskočí a odpovídající TDA nastaví na výchozí hodnotu. V tab. 6.3 jsou uvedeny výchozí hodnoty používané u radiového přenosu. [7]

Tab. 6.3 Výchozí nastavení rádiového přenosu

Kroky	Proměnná TDA	Výchozí nastavení
0. Skupina přijímačů	není	prázdná skupina přijímačů (stejné chování podle vzoru dra_no_rxgroup)
1. Zpoždění přenosu	OPC_TDA_RA_TX_DELAY	0 sekund
2. Rezervace linky	OPC_TDA_RA_CLOUSURE	OPC_TRUE
3. Výběr kanálu	OPC_TDA_RA_MATCH_STATUS	OPC_TDA_RA_MATCH_VALID
4. Zisk vysílací antény	OPC_TDA_RA_TX_GAIN	0 dB

Kroky	Proměnná TDA	Výchozí nastavení
5. Zpoždění vysílaní	OPC_TDA_RA_START_PROPDEL	0 sekund
	OPC_TDA_RA_END_PROPDEL	0 sekund
6. Zisk přijímací antény	OPC_TDA_RA_RX_GAIN	0 dB
7. Síla přijatého signálu	OPC_TDA_RA_RCVD_POWER	(výkon vysílače) * (zisk vysílací antény) (počítané podle vzorce OPC_TDA_RA_TX_POWER* $10^{OPC\_TDA\_RA\_TX\_GAIN}$* $10^{OPC\_TDA\_RA\_RX\_GAIN}$)
8. Interferenční šum	není	žádné rušení
9. Šum pozadí	OPC_TDA_RA_BKGNOISE	0 Watt
10. Poměr signál - šum	OPC_TDA_RA_SNR	1000 dB (libovolné velmi vysoké hodnoty, bez rušení)
11. Bitová chybovost	OPC_TDA_RA_BER	0
12. Alokace chyb	OPC_TDA_RA_NUM_ERRORS	0
	OPC_TDA_RA_ACTUAL_BER	OPC_TDA_RA_BER
13. Korekce chyb	OPC_TDA_RA_PK_ACCEPT	OPC_TRUE

Výchozí hodnoty mohou být změněny nastavením odpovídajících proměnných, jejichž seznam je v tab. 6.4. Tyto proměnné lze definovat prostřednictvím **Preferences Editoru**, který lze vyvolat položkou **Preferences** v menu **Edit** po otevření okna **Configure/Run Discrete Event Simulation (Advanced)** v menu **DES** projektu v OM (obr. 6.1). [7]



Obr. 6.1 Změna výchozí hodnoty TDA atributu

Tab. 6.4 Tag výchozí hodnoty TDA

Krok přenosu	Proměnná TDA	Tag výchozí hodnoty
0. Skupina přijímačů	není	není
1. Zpoždění přenosu	OPC_TDA_RA_TX_DELAY	psnone_tda.ra.txdel.tx_delay
2. Rezervace linky	OPC_TDA_RA_CLOUSURE	psnone_tda.ra.closure.closure
3. Výběr kanálu	OPC_TDA_RA_MATCH_STATUS	psnone_tda.ra.chanmatch.match_status
4. Zisk vysílací antény	OPC_TDA_RA_TX_GAIN	psnone_tda.ra.tagain.tx_gain
5. Zpoždění vysílaní	OPC_TDA_RA_START_PROPDEL	psnone_tda.ra.propdel.start_propdel
	OPC_TDA_RA_END_PROPDEL	psnone_tda.ra.propdel.end_propdel
6. Zisk přijímací antény	OPC_TDA_RA_RX_GAIN	psnone_tda.ra.ragain.rx_gain
7. Síla přijatého signálu	OPC_TDA_RA_RCVD_POWER	psnone_tda.ra.power.rcvd_power
8. Interferenční šum	OPC_TDA_RA_BKGNOISE	psnone_tda.ra.bkgnoise.bkgnoise
9. Šum pozadí	není	není
10. Poměr signál – šum	OPC_TDA_RA_SNR	psnone_tda.ra.snr.snr
11. Bitová chybovost	OPC_TDA_RA_BER	psnone_tda.ra.ber.ber
12. Alokace chyb	OPC_TDA_RA_NUM_ERRORS	psnone_tda.ra.error.num_error
	OPC_TDA_RA_ACTUAL_BER	psnone_tda.ra.error.actual_ber
13. Korekce chyb	OPC_TDA_RA_PK_ACCEPT	psnone_tda.ra.ecc.pk_accept

6.1 Procedury jádra pracující s TDA

Výsledky zprostředkovávané prostřednictvím TDA je třeba během přenosu v jednotlivých modelech opakovaně aktualizovat nebo číst.

Pro přiřazení hodnot TDA lze použít následující procedury jádra. *op_td_set_int()*, *op_td_set_int64()*, *op_td_set_dbl()* a *op_td_set_ptr()*. Naopak pro čtení hodnot lze použít procedury *op_td_get_int()*, *op_td_get_int64()*, *op_td_get_dbl()* a *op_td_get_ptr()*. Podle poslední části názvu procedury lze pracovat s datovými typy 32 i 64 bitový integer, double a s ukazateli. Pro rychlé inkrementování hodnot lze použít procedury *op_td_increment_int()*, *op_td_increment_int64()* a *op_td_increment_dbl()*. A nakonec procedura *op_td_is_set()* určuje, zda zadanému TDA byla přiřazena hodnota.

Hlavními argumenty procedur jsou *pkptr* a *tda_index*. *Pkptr* je ukazatel na aktuální paket v kroku přenosu a *tda_index* je celočíselný index TDA. V případě nastavení hodnoty je třetím argumentem tato hodnota s příslušným datovým typem. [7]

OP_TD_GET_DBL()

```
op_td_get_dbl (pkptr, tda_index)
```

Procedura umožňuje získat jednu hodnotu TDA typu *double*, která je spojena s každým paketem, který prochází daným krokem přenosu.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha k

```
rx_bw = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_BW).
```

OP_TD_GET_INT()

```
op_td_get_int (pkptr, tda_index).
```

Procedura umožňuje získat jednu hodnotu TDA typu 32 bitový *integer*, která je spojena s každým paketem, který prochází daným krokem přenosu.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha i

```
rx_ch_obid = op_td_get_int (pkptr, OPC_TDA_RA_RX_CH_OBJID).
```

OP_TD_GET_INT64()

```
op_td_get_int64 (pkptr, tda_index).
```

Procedura umožňuje získat jednu hodnotu TDA typu 64 bitový *integer*, která je spojena s každým paketem, který prochází daným krokem přenosu.

OP_TD_GET_PTR()

```
op_td_get_ptr (pkptr, tda_index).
```

Procedura umožňuje získat jeden ukazatel na hodnotu TDA, který je spojena s každým paketem, který prochází daným krokem přenosu. Typické využití ukazatelů je ve spojení s komplexními informacemi, jako jsou např. tabulky.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha m

```
modulation_table = op_td_get_ptr (pkptr, OPC_TDA_RA_RX_MOD).
```

OP_TD_INCREMENT_DBL()

```
op_td_increment_dbl (pkptr, tda_index, value)
```

Procedura se používá při změně hodnoty TDA. Poskytuje rychlou alternativu k využití *op_td_get_dbl()* a *op_td_set_dbl()*.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha j

```
op_td_increment_dbl (pkptr_arriv, OPC_TDA_RA_NOISE_ACCUM, rev_rcvd_power).
```

OP_TD_INCREMENT_INT()

```
op_td_increment_int (pkptr, tda_index, value)
```

Procedura se používá při změně hodnoty TDA. Poskytuje rychlou alternativu k využití *op_td_get_int()* a *op_td_set_int()*.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha j `op_td_increment_int (pkptr_prev, OPC_TDA_RA_NUM_COLLS, 1)`.

OP_TD_INCREMENT_INT64()

```
op_td_increment_int64 (pkptr, tda_index, value)
```

Procedura se používá při změně hodnoty TDA. Poskytuje rychlou alternativu k využití *op_td_get_int64()* a *op_td_set_int64()*.

OP_TD_IS_SET()

```
op_td_is_set (pkptr, tda_index)
```

Procedura se používá ke zjištění, zda TDA má nastavenu hodnotu. To je užitečné v případě, když kroky přenosu závisejí na speciální vlastnosti přítomné v paketu. Pokus o přístup k atributu, který není nastaven, způsobí chybu simulace. Proto, pokud lze očekávat některou z těchto možností, měl by být TDA testován na nastavenou hodnotu.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha o `op_td_is_set (pkptr, OPC_TDA_RA_ND_FAIL)`.

OP_TD_SET_DBL()

```
op_td_set_dbl (pkptr, tda_index, value)
```

Procedura umožňuje nastavit jednu hodnotu TDA typu double, která je spojena s každým paketem, který prochází daným krokem přenosu.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha d `op_td_set_dbl (pkptr, OPC_TDA_RA_RCVD_POWER,pow (10.0, tmm_model_path_loss_dB / 10.0))`.

OP_TD_SET_INT()

```
op_td_set_int (pkptr, tda_index, value)
```

Procedura umožňuje nastavit jednu hodnotu TDA typu 32 bitový *integer*, která je spojena s každým paketem, který prochází daným krokem přenosu.

Příklad kompletního zdrojového kódu této procedury je uveden v příloha d `op_td_set_int (pkptr, OPC_TDA_RA_CLOSURE, OPC_FALSE)`.

OP_TD_SET_INT64()

```
op_td_set_int64 (pkptr, tda_index, value)
```

Procedura umožňuje nastavit jednu hodnotu TDA typu 64 bitový *integer*, která je spojena s každým paketem, který prochází daným krokem přenosu.

OP_TD_SET_PTR()

```
op_td_set_ptr (pkptr, tda_index, value)
```

Procedura umožňuje nastavit jeden ukazatel na hodnotu TDA, který je spojen s každým paketem, který prochází daným krokem přenosu. Typické využití ukazatelů je ve spojení s komplexními informacemi, jako jsou např. tabulky.

7 VYTVOŘENÍ UŽIVATELSKÉHO TDA

Index TDA je celé číslo, které specifikuje uvažovaný TDA. Názvy předdefinovaných TDA představují právě tento index a poslední nejvyšší rezervovaný index je k dispozici v symbolické konstantě *OPC_TDA_RA_MAX_INDEX*.

Vytvoření uživatelského TDA spočívá v nastavení indexu na hodnotu vyšší, než na které je umístěn poslední předdefinovaný TDA.

Novému TDA lze přidělit název direktivou jazyka C++ *#define*, není to však nutné, lze totiž pracovat pouze s inkrementovanou symbolickou konstantou *OPC_TDA_RA_MAX_INDEX*.

S takto nově definovaným TDA lze pracovat stejně jako s jakýmkoli jiným předdefinovaným TDA.

V této části práce jsou vytvořeny tři nové TDA a prověřena jejich funkčnost v simulaci. Potřebné úpravy zdrojových kódů byly provedeny pouze ve zdrojových kódech výchozích modelů radiového přenosu.

První uživatelský TDA se jmenuje *OPC_TDA_RA_TX_SILA*. Tento název je definován pomocí direktivy

```
#define OPC_TDA_RA_TX_SILA OPC_TDA_RA_MAX_INDEX + 1
```

a je potřeba ji uvést v každém modelu, ve kterém chceme s takto pojmenovaným TDA pracovat.

V modelu *dra_tagain.ps.c* je síla přijatého signálu uložena do nového TDA jako logaritmické vyjádření přijímaného výkonu tx_power v datovém typu double.

```
op_td_set_dbl (pkptr, OPC_TDA_RA_TX_SILA, 10.0*log10(tx_power));
```

Následující řádek vyjadřuje tutéž definici bez použití jména TDA

```
op_td_set_dbl (pkptr, OPC_TDA_RA_MAX_INDEX + 1, 10.0*log10(tx_power));
```

V modelu *dra_power.ps.c* je nový TDA načten do proměnné tx_sila:

```
tx_sila = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_SILA);
```

převeden na celočíselný formát:

```
ftx_sila = tx_sila + 0.5;
```

a vypsán do simulační konzole:

```
printf(„Sila prijimaciho uzlu = %d dB", ftx_sila);
```

Druhý a třetí uživatelský TDA nemají definovaný název a jsou určeny k číslování paketů.

TDA „OPC_TDA_RA_MAX_INDEX + 2“ zaznamenává počet již odeslaných paketů vyslaných vysílacím uzlem včetně právě vyslaného.

TDA „OPC_TDA_RA_MAX_INDEX + 3“ zaznamenává počet paketů vyslaných rušícím uzlem včetně právě vyslaného.

V modelu dra_txdel.ps.c jsou tyto TDA podmíněně aktualizovány z globálních proměnných Packet_nr_tx a Packet_nr_jam.

```
CISLO PAKETU VYSILACIHO UZLU = 27
      Zpozdeni vysilaciho uzlu= 1 s

CISLO PAKETU RUSICIHO UZLU = 27
      Zpozdeni rusiciho uzlu= 1 s

STRANA PRIJIMACE
snr_time = 35s    SNR = -9.474722   EFF_SNR = 0.422279      BER = 0.074032
Sila vysilaciho uzlu = 0 dB
snr_time = 36s    SNR = -9.458735   EFF_SNR = 0.438265      BER = 0.074032
snr_time = 36s    SNR = 94.477676   EFF_SNR = 104.374680    BER = 0.000000

CISLO PAKETU VYSILACIHO UZLU = 28
      Zpozdeni vysilaciho uzlu= 1 s

CISLO PAKETU RUSICIHO UZLU = 28
      Zpozdeni rusiciho uzlu= 1 s

STRANA PRIJIMACE
snr_time = 36s    SNR = -9.469545   EFF_SNR = 0.427455      BER = 0.074032
Sila vysilaciho uzlu = 0 dB
snr_time = 37s    SNR = -9.453531   EFF_SNR = 0.443469      BER = 0.074032
snr_time = 37s    SNR = 94.493690   EFF_SNR = 104.390694    BER = 0.000000

CISLO PAKETU VYSILACIHO UZLU = 29
      Zpozdeni vysilaciho uzlu= 1 s

CISLO PAKETU RUSICIHO UZLU = 29
      Zpozdeni rusiciho uzlu= 1 s

STRANA PRIJIMACE
snr_time = 37s    SNR = -9.464355   EFF_SNR = 0.432645      BER = 0.074032
Sila vysilaciho uzlu = 0 dB
```

Obr. 7.1 Ukázka výpisu simulační konzole

Na začátku výpisu simulační konzole (obr. 7.1) je pro každý paket vysílacího a rušícího uzlu vypsáno číslo paketu a zpoždění přenosu definující čas mezi vysláním prvního bitu a ukončením příjmu posledního bitu paketu. Čísla paketů jsou počítány v rámci nově vytvořených TDA. Zpoždění 1s vychází z nastavení rychlosti přenosových kanálů. Dále jsou vypsány parametry platných paketů vysílacího uzlu na straně přijmače. Prvním parametrem je čas výpočtu SNR v rámci simulovaného času přenosu, druhým parametrem je hodnota SNR v dB, třetím efektivní hodnota SNR v dB, což je hodnota SNR zahrnující zpracování zisku přijímacího systému a nakonec hodnota BER. Tyto hodnoty jsou počítány pro každý paket celkem třikrát pro jednotlivé segmenty paketu s konstantní hodnotou BER, jak je vysvětleno v popisu kroku. Pro první dva segmenty paketu je hodnota SNR záporná, což svědčí o tom, že v přijímaném signálu převažuje šumová

složka. Ve třetím segmentu paketu je hodnota SNR kladná, což nasvědčuje tomu, že výrazně převažuje platný signál. Tyto výsledky potvrzuje i hodnota BER, která je narozdíl od prvních dvou ve třetím segmentu nulová. Mezitím je vypsána síla vysílacího uzlu v dB jakožto nově vytvořeného TDA OPC_TDA_RA_TX_SILA. Tato hodnota je 0dB což odpovídá použité všesměrové anténě.

ZÁVĚR

Cílem této diplomové práce bylo seznámení se a popis možností modelování bezdrátových technologií v simulačním prostředí OPNET Modeler následované sérií simulací pro ověření vlivu parametrů bezdrátového přenosu na jeho kvalitu. Dále popis obecného modelu vysílače a přijímače v tomoto prostředí a metod výpočtů a přenosů parametrů v rámci simulace.

Nejdříve byl nakonfigurován model v základní topologii obsahující vysílací a přijímací modul s všesměrovými anténami a pohyblivý rušící modul, který se od vysílacího modulu liší výkonem a druhem modulace. Z výsledů je patrné, že se zkracující se vzdáleností rušícího uzlu od uzlu vysílače a přijímače se zvyšuje bitová chybovost systému. Během simulace bylo přeneseno 1418 paketů, z nichž bylo 13 přijato a 1405 odmítnuto.

Následně byla síť různými způsoby pozměněna a to typem antény, typem modulace a topologií sítě. Vedle všesměrové antény byla použita anténa směrová a všechny varianty simulovány s použitím modulace BPSK, PSK8 a QAM64.

Ze získaných výsledků je patrné, že použití směrové antény vede ke zvýšení odolnosti proti rušení rušícím uzlem. V grafech vyjadřujících průběh bitové chybovosti na čase je vidět, že bitová chybovost se výrazně zvýší pouze v případě, kdy rušící uzel prochází spojnicí uzlů vysílače a přijímače. Tak se počet správně přijatých paketů zvýšil zhruba na polovinu všech vyslaných paketů, což odpovídá podílu vysílacího uzlu v celkovém počtu paketů. Dále se potvrdilo, že modulace BPSK odolává rušení lépe než vícestavové modulace PSK8 a QAM64, kde při použití všesměrové antény nebyl správně doručen ani jediný paket. Při použití směrové antény byla správně doručena opět zhruba polovina ze všech vyslaných paketů, ovšem opět méně, než při použití modulace BPSK.

Změna topologie sítě spočívala ve změně umístění uzlů a změně trajektorie pohyblivého uzlu. Použita byla všesměrová i směrová anténa pro přenos využívající již jen modulaci BPSK. I zde se potvrdilo stejně jako v modelovém případě, že bitová chybovost narůstá se zkracující se vzdáleností rušícího uzlu a výhody směrové antény. Nejvyššího počtu správně přijatých paketů bylo dosaženo topologii 2, kde rušící uzel není pohyblivý a je maximálně vzdálen od vysílače. Pohyblivým uzlem je přijímač. Při použití všesměrové antény bylo správně přijato 154 a v případě směrové antény 709 z celkového počtu 1418 paketů.

Funkce simulací přenosů spočívá v rozdělení přenosového kanálu na 14 kroků, z nichž každý je realizován modelem tvořeným zdrojovým kódem v programovacím jazyce C++. Vzájemná komunikace a přenos dat mezi jednotlivými modely spočívá v definici speciálních datových jednotek TDA, zpravidla vázaných na pakety přenosu. Tyto datové jednotky jsou proměné parametry zvoleného datového typu, které jsou během přenosu v příslušných modelech postupně nastavovány a opakovaně používány. Výpočet jejich hodnot je popsán v rámci kroků přenosu a zdrojové kódy jsou k dispozici v přílohách práce.

Prostřednictvím procedur umožňujících práci s TDA byly vytvořeny tři nové TDA a jejich hodnoty byly spolu s vybranými hodnotami dalších předdefinovaných TDA vypisovány do okna simulační konzole pomocí příkazů jazyka C++. Tyto úpravy modelů a simulace byly realizovány pro základní simulační topologii a uzly.

Nejdříve je vypsáno číslo paketu a zpoždění vysílání uzlu nejdříve pro vysílací a potom rušící uzel. Čísla paketů jsou načítána z nově vytvořených TDA a zpoždění vysílání 1s je definováno výchozím nastavením modelu. Následně jsou na straně přijímače vypsány parametry

přenosu, konkrétně pak čas výpočtu SNR (snr_time), SNR, efektivní SNR zahrnující i zpracování zisku přijímače a bitová chybovost BER. Tyto parametry jsou počítány pro jeden paket 3 krát, vždy pro daný segment paketu, pro který zůstává hodnota SNR konstantní. Počet těchto segmentů je dán činností ostatních vysílačů. Mezitím je vypsána síla vysílacího uzlu v dB jakožto hodnota nově vytvořeného TDA. Hodnota 0dB potvrzuje podmínku pro speciální případ použití všesměrové antény, kdy výpočet není potřeba.

Všechny operace s TDA pracovaly bez problémů. Přenos hodnot mezi modely u nově vytvořených TDA pracoval bez problémů a jejich hodnoty včetně dalších předdefinovaných TDA vypsané v simulační konzoli odpovídaly předpokladům.

SEZNAM LITERATURY

[1] *Archiv článku a přednášek Jiřího Peterky: Bezdrátové přenosy* [online].
Dostupné z WWW: http://www.earchiv.cz/a96/a647k150.php3

[2] *Faktory působící na šíření radiových vln* [online].
Dostupné z WWW: http://fei1.vsb.cz/kat420/vyuka/fei/sireni_vln/teze/otazka_11.pdf

[3] GAST, M. *802.11 Wireless Networks: The Definitive Guide*. Second Edition. Sebastopol : Media, 2005.

[4] *Historie antén a bezdrátové komunikace* [online]. 2010
Dostupné z WWW: http://www.urel.feec.vutbr.cz/~raida/multimedia/cz/10-1-A.pdf

[5] KOCOUREK, J., NOVÁK, J. Přenos informace. Skriptum. České vysoké učení v Praze. Praha, 2006.

[6] *Odraz, lom a ohyb vlnění* [online].
Dostupné z WWW:
http://www.realisticky.cz/ucebnice/02%20Fyzika/03%20Kmitav%C3%BD%20pohyb%20a%20mechanick%C3%A9%20vln%C4%9Bn%C3%AD/02%20Mechanick%C3%A9%20vln%C4%9Bn%C3%AD/05%20Odraz,%20lom%20a%20ohyb%20vln%C4%9Bn%C3%AD.pdf

[7] OPNET Technologies, OPNET Modeler Product Documentation Release 16.1, 2010

[8] PECHAČ, P., ZVÁNOVEC, S. *Základy šíření vln*. 1. vydání. BEN – technická literatura. Praha, 2007.

[9] PROKEŠ, A. *Rádiové a mobilní komunikace*. Elektronické skriptum. VUT Brno.Brno, 2005.

[10] *Vznik elektromagnetického vlnění* [online].
Dostupné z WWW: http://elmag.sps.sweb.cz/1.htm

[11] Wikipedia. *Frequency-shift keying*. [online]. 2008. Dostupné z WWW: http://en.wikipedia.org/wiki/Frequency-shift_keying.

[12] Wikipedia. *Hammingův kód.* [online]. 2008. Dostupné z WWW: http://cs.wikipedia.org/wiki/Hamming%C5%AFv_k%C3%B3d.[13] ŽALUD, V. *Moderní radioelektronika*. 1. vydání. BEN – technická literatura. Praha, 2000.

SEZNAM POUŽITÝCH ZKRATEK

-	ASK	Amplitude-Shift Keying
-	BER	Bitová Chybovost
-	BPSK	Binary-Phase Shift Keying
-	CDMA	Code Divition Multiple Access
-	FSK	Frequency-Shift Keying
-	FSPL	Free-Space Path Loss
-	FTP	File Transfer Protocol
-	GSM	Global System for Mobile Communication
-	HTTP	Hypertext Transfer Protocol
-	IEEE	Institute of Electrical and Electronics Engineers
-	LAN	Local Area Network
-	LTE	3GPP Long Term Evolution
-	ODB	OPNET Debugger
-	OM	OPNET Modeler
-	PSK	Phase-Shift Keying
-	PSK8	8 Phase-Shift Keying
-	QAM64	Quadrature amplitude modulation
-	SNR	Poměr Signál-Šum
-	TDA	Transmission Data Attribute
-	TMM	Terrain Modeling Module
-	UMTS	Universal Mobile Telecommunication System
-	VoIP	Voice over Internet Protocol
-	WiMAX	Worldwide Interoperability for Microwave Access
-	WLAN	Wireless Local Area Network

SEZNAM PŘÍLOH

PŘÍLOHA A Kód zaměřovacího procesoru

```
Objid subnet_id,                /* identifikator podsiti */
      tx_node_id,               /* identifikator prijimace */
      rx_node_id,               /* identifikator vysilace */
      rx_ant_id;                /* identifikator anteny prijimace */
double altitude,                /* nadmorska vyska */
      latitude,                 /* zemepisna sirka */
      longitude,                /* zemepisna delka */
      x_pos,                    /* x-pozice vysilace v podsiti */
      y_pos,                    /* y-pozice vysilace v podsiti */
      z_pos;                    /* z-pozice vysilace v podsiti */
Compcode comp_code;             /* kod dokonceni procedury */

/* Tento radek uklada ID rodicovskeho uzlu  do promenne rx_node_id. */
/* Funkce op_id_self() urcuje ID objektu modulu. Funkce op_topo_parent() pouziva ID */
/* objektu k urceni ID rodicovskeho objektu. */
rx_node_id = op_topo_parent (op_id_self ());

/*Tento radek uklada ID subnetu do promenne subnet_id.*/
subnet_id = op_topo_parent (rx_node_id);

/*Tento radek uklada ID vysilaciho uzlu do promenne tx_node_id, k tomu je pouzito*/
/*jmeno vysilaciho uzlu jako argument funkce op_id_from_name(), ktere urcuje ID*/
/*objektu z poskytovanych parametru: ID rodicovskeho objektu, typ a jmeno objektu*/
tx_node_id = op_id_from_name (subnet_id, OPC_OBJTYPE_NDFIX, "tx");

/*Tento radek pouziva funkce op_ima_obj_pos_get() aktualizuje hodnoty polohy*/
/*vysilaciho uzlu. Zaroven prevadi relativni polohu vysilaciho uzlu na geocentricke*/
/*souradnice. Souradnice x,y,z zde nebudou vyuzity ale procedura je vyzaduje jako*/
/*argumenty.*/
comp_code = op_ima_obj_pos_get (tx_node_id,
             &latitude, &longitude, &altitude, &x_pos, &y_pos, &z_pos);

/*Porovnani hodnoty comp_code se symbolickou konstantou OPC_COMPCODE_SUCCESS a*/
/*OPC_COMPCODE_FAILURE urcuje spravne nebo chybne provedeni predchozi operace. Tento*/
/*radek testuje comp_code  a při detekci chyby vola funkci op_sim_end() k okamzitemu*/
/*ukonceni simulace a zobrazeni chybove zpravy. */
if (comp_code == OPC_COMPCODE_FAILURE)
      op_sim_end ("get attributes failed", "", "", "");

/*Tento radek prirazuje ID anteny v prijimacim uzlu do promenne rx_ant_id. */
/*Je pouzito jmeno anteny jako argument funkce op_id_from_name().*/
rx_ant_id = op_id_from_name (rx_node_id, OPC_OBJTYPE_ANT, "ant_rx");

/*Zbyvajici radky pouzivaji funkci op_ima_obj_attr_set_dbl() k nastavovani cilove*/
/*polohy anteny geocentrickymi souradnicemi (zemepisna delka a sirka, a nadmorska*/
/*vyska) s hodnotami vyvolanymi uzlem vysilace. Jestlize comp_code oznaci chybu, */
/*simulace je ihned ukoncena. */
comp_code = op_ima_obj_attr_set_dbl (rx_ant_id, "target altitude", altitude+1.5);
if (comp_code == OPC_COMPCODE_FAILURE)
      op_sim_end ("Set target altitude failed.", "", "", "");

comp_code = op_ima_obj_attr_set_dbl (rx_ant_id, "target latitude", latitude);
if (comp_code == OPC_COMPCODE_FAILURE)
      op_sim_end ("Set target latitude failed.", "", "", "");

comp_code = op_ima_obj_attr_set_dbl (rx_ant_id, "target longitude", longitude);
if (comp_code == OPC_COMPCODE_FAILURE)
      op_sim_end ("Set target longitude failed.", "", "", "");
```

PŘÍLOHA B Kód modelu 0.kroku dra_rxgroup

```
/* dra_rxgroup.ps.c */
/* Vychozi model kroku "skupina prijimacu" radioveho prenosu      */
/* Tento model nastavi stavove informace o prijimacich kanalech   */
/* pro pouziti v krocich sily prijateho signalu a korekce chyb.   */
/* Pokud neni treba, aby model tyto informace poskytl, potom      */
/* je mozno pouzit "*_no_rxstate" verzi modelu rxgroup, power a ecc*/

/*************************************/
/*      Copyright (c) 1993-2009  )      */
/*     by OPNET Technologies, Inc.      */
/*     (A Delaware Corporation)         */
/*     7255 Woodmont Av., Suite 250     */
/*     Bethesda, MD 20814, U.S.A.       */
/*     All Rights Reserved.             */
/*************************************/

#include "opnet.h"
#include "dra.h"

#if defined (__cplusplus)
extern "C"
#endif

int
dra_rxgroup_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Objid PRG_ARG_UNUSED(tx_obid), Objid
rx_obid)
       {
       DraT_Rxch_State_Info*     rxch_state_ptr;

  /** Urceni moznosti komunikace mezi danymi vysilacimi**/
  /** a prijimacimi uzly. Dale vytvoreni a inicializace**/
  /** stavovych informaci prijimacich kanalu pro pouziti **/
  /** v dalsich krocich radioveho prenosu behem simulace **/
       FIN_MT (dra_rxgroup (tx_obid, rx_obid));

  /* Pokud je jiz hotovo, inicializace stavovych informaci */
  /* prijimacich kanalu   */
       if (op_ima_obj_state_get (rx_obid) == OPC_NIL)
             {#if defined (OPD_PARALLEL)

    /* Stavove informace prijimacich kanalu neexistuji.    */
    /* Pred pokracovanim uzamci globalni mutex.      */
             op_prg_mt_global_lock ();

    /* Nova kontrola, protoze jine vlakno muze jiz   */
    /* stavove informace nastavovat.    */
             if (op_ima_obj_state_get (rx_obid) == OPC_NIL)
             {#endif /* OPD_PARALLEL */

      /* Vytvoreni a nastaveni stavovych informaci prijimaciho kanalu   */
      /* Stavove informace jsou vyuzity dalsimi kroky prenosu     */
      /* k efektivnimu pristupu a aktualizivani specifickych dat. */
                  rxch_state_ptr = (DraT_Rxch_State_Info *)
                        op_prg_mem_alloc (sizeof (DraT_Rxch_State_Info));
                  rxch_state_ptr->signal_lock = OPC_FALSE;
                  op_ima_obj_state_set (rx_obid, rxch_state_ptr);
                  #if defined (OPD_PARALLEL) }

    /* Odemceni globalniho mutexu.            */
             op_prg_mt_global_unlock ();
             #endif /* OPD_PARALLEL */ }

  /* Ve vychzim nastaveni jsou vsechny prijimaci kanaly    */
  /* povazovany za potencialni cile signalu */
       FRET (OPC_TRUE)}
```

PŘÍLOHA C Kód modelu 1.kroku dra_txdel

```
/* dra_txdel.ps.c */
/* Vychozi model kroku "zpozdeni prenosu" radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009       */
/*           by OPNET Technologies, Inc.     */
/*           (A Delaware Corporation)        */
/*     7255 Woodmont Av., Suite 250          */
/*      Bethesda, MD 20814, U.S.A.           */
/*           All Rights Reserved.            */
/*****************************************/

#include "opnet.h"

#if defined (__cplusplus)
extern "C"
#endif
void
dra_txdel_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       OpT_Packet_Size     pklen;
       double              tx_drate, tx_delay;

  /** Vypocet zpozdeni radioveho prenosu spojeneho s prenosem paketu    **/
       FIN_MT (dra_txdel (pkptr));

  /* Nacteni prenosove rychlosti kanalu . */
       tx_drate = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_DRATE);

  /* Nacteni delky paketu. */
       pklen = op_pk_total_size_get (pkptr);

  /* Vypocet casu ptrebneho k dokonceni prenosu paketu. */
       tx_delay = pklen / tx_drate;

  /* Ulozeni vysledku zpozdeni prenosu do prislusneho TDA. */
       op_td_set_dbl (pkptr, OPC_TDA_RA_TX_DELAY, tx_delay);

       FOUT
       }
```

PŘÍLOHA D Kód modelu 2.kroku dra_closure

```
/* dra_closure.ps.c */
/* Model rezervace linky s podporou TMM. Tento krok urcuje, zda je      */
/* omunikace mezi uzly mozna. Pri spusteni behem TMM simulace bude      */
/* pro urceni dostupnosti spojeni a ztraty vykonu signalu vlivem sireni */
/* vlneni vyuzit model TMM. Ztrata vykonu bude pozdeji vyuzita pri  */
/* vypoctu sily prijateho signalu */

/***************************************/
/*       Copyright (c) 1993-2009       */
/*      by OPNET Technologies, Inc.     */
/*           (A Delaware Corporation)   */
/*     7255 Woodmont Av., Suite 250     */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.*/
/***************************************/

#include <math.h>
#include <string.h>
#include "opnet.h"
#include "closure_support.h"

/* Tento krok pracuje ve trech modech:               */
/* rezim #1 Cesta prenosu nebude nikdy blokovana.    */
/* OPC_TDA_RA_CLOSURE bude vzdy nastaven na OPC_TRUE.      */
/* Toto je vychozi mod bezdratoce site, kdy TMM neni aktivni     */
/* rezim #2 Vychzi prostor pro sireni vlneni je pro vsechny */
/* prenosy kulovy model Zeme s algoritmem Line-of-Sight    */
/* rezim #3 s vyuzitim TMM modelu      */

/****** Vycet definice typu.     ******/
typedef enum DraT_Closure_Method
      {
      DraC_Line_Of_Sight_Never_Occluded,
      DraC_Earth_Line_Of_Sight,
      DraC_Terrain_Modeling
      } DraT_Closure_Method;

/***** Globalni promenne               *****/
/* Promenne jsu definovany puze jednou a vyuzivany po celu dobu simulace             */
/* Tato promenna bude obsahovat blokovani kanalu po celou dobu simulace */
/* Jeji mozne hodnoty odpovidaji hodnotam definovanym v    DraT_Closure_Method */
/* To je definvano z Earth_Line_Of_Sight. Nicmene jeji vysledna hodnota bude   */
/* nastavena z tmm_closure_init() */

static DraT_Closure_Method             DraS_Active_Closure_Method =
DraC_Earth_Line_Of_Sight;

/* Jestlize pracujeme v rezimu 3, zde je TMM model pro vypocet ztaty sireni */
static TmmT_Propagation_Model *        DraS_Closure_Prop_Model_Ptr = OPC_NIL;

/* Jestlize pracujeme v rezimu 3, zde je jmeno TMM modelu*/
static const char *                          DraS_Closure_Tmm_Prop_Model_Name;

/* V rezimu tmm_verbose jsou pripozadavku modelu zpravy popisuji        */
/* prenos paketu zaznamenavany.  */
Log_Handle                                        DraS_TMM_Verbose_Log_H;

/***** Prototypy funkci.  *****/

static void  tmm_closure_init                        (OP_SIM_CONTEXT_ARG_OPT);
static void dra_non_tmm_closure_method_set    (OP_SIM_CONTEXT_ARG_OPT);
static void  tmm_model_closure_calc                  (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet
* pkptr);

/***** konstanty *****/
/* Prahova hodnota v dB. Ztraty sirenim prekracujici tuto hranici */
```

```
/* zpusobi nemoznost navazat spojeni */
#define LOSS_CUTOFF_THRESHOLD_DB -140.0

/* Jmeno globalniho atributu pouite v modelu k vyberu mezi dvema ne TMM  */
/* metodami pro urceni moznosti spojeni. */
/*Tento atribut je definovan v wlan_dispatch modelu sireni*/
#define CLOSURE_METHOD_GLOBAL_ATTRIBUTE_NAME  "Closure Method (non-TMM)"

/***** Procedury prenosu *****/

#if defined (__cplusplus)
extern "C"
#endif
void
dra_closure_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       static int   closure_initialized = 0;

       FIN_MT (dra_closure (pkptr));

  /* Pri pouziti paralelniho jadra simulace muze tento bod algoritmu    */
  /* obdrzet nekolik vypocetnich vlaken ve stejnou dobu. Je potreba     */
  /* zajistit, aby tmm_closure byl volan pouze jednou. Tento pozadavek  */
  /* umoznuje jistou volnost. Closure_initialized muze byt nastaveno na 1,     */
  /* zatimco jine vlakno cte hodnotu. Tak jine vlakno obdrzi hodnotu    */
  /* (0 nebo 1, nebo jiny sled bitu). Pokud obdrzi 1 nebo chybu, test selze */
  /* Ale protoze vysledek je jiz dokoncen, vse je v poradku */
  /* Pokud obdrzi 0, je snaha vytvorit mutex a uzamknout jej.     */
  /* Muze se objevit nekolik volani    op_prg_mt_mutex_create, */
  /* ale vsechny vrati stejny mutex ukazatel  */
  /* Zapis stejne hodnoty do globalni promenne by mel vydrzet neatomicitu*/
       if (!closure_initialized)
              {
#if defined  (OPD_PARALLEL)
              op_prg_mt_global_lock ();
#endif

    /* Opetovna kontrola promenne. Jestlize jine vlakno jiz vytvorilo  */
    /* mutex a byl jiz inicializovan, hodnota bude nastavena na 1 a  */
    /* test selze */
              if (!closure_initialized)
                     {
                     /* This function will determine the behavior for           */
                     /* closure by setting static variables                            */
                     /* (DraS_Active_Closure_Method,                                   */
                     /* DraS_Closure_Prop_Model_Ptr).                                  */

      /* Tato funkce urcuje chovani blokovani nastavenim statickych */
      /* promennych */
      /* (DraS_Active_Closure_Method, */
       /* DraS_Closure_Prop_Model_Ptr).*/
                     tmm_closure_init (OP_SIM_CONTEXT_PTR_OPT);

       /* Bude provedena pouze jedna inicializace. Resetujeme  */
      /* priznak abychm zajistili, ze jine vlakno neprevezme inicializaci */
      /* V nejhorsim pripade budou mit dalsi vlakna stejny mutex a budou  */
      /* smerovany na blokaci */
                     closure_initialized = 1;
                     }
#if defined  (OPD_PARALLEL)
              op_prg_mt_global_unlock ();
#endif
              }
  /* Tento krok ma tri rezimy provadeni. */
  /* Podrobnejsi komentar u definide */
       switch (DraS_Active_Closure_Method)
              {
              case DraC_Line_Of_Sight_Never_Occluded:
```

```
      /* Rezim 1: Cesta prenosu nebude nikdy blokvana */
      /* nastaveni OPC_TDA_RA_CLOSURE na OPC_TRUE u vsech prenosu */
                  op_td_set_int (pkptr, OPC_TDA_RA_CLOSURE, OPC_TRUE);
            break;
            case DraC_Earth_Line_Of_Sight:

      /* Rezim 2: Zakladni kulovy model Zeme*/
                  generic_earth_LOS_closure (OP_SIM_CONTEXT_PTR_OPT_COMMA pkptr);
            break;
            case DraC_Terrain_Modeling:

        /* Rezim 3: Vyuziti TMM modelu sireni*/
                  tmm_model_closure_calc (OP_SIM_CONTEXT_PTR_OPT_COMMA pkptr);
            break;
            }
      FOUT
      }

/***** Podpurne funkce *****/

static void
tmm_closure_init (OP_SIM_CONTEXT_ARG_OPT)
      {
      int                        using_tmm;
      Log_Handle          tmm_problem_log_handle;
      int                        load_successful;
      char                line0_buf [512];
      char                line1_buf [512];

  /** Tato funkce je volana jednou na zacatku simulace     **/
  /** Zde jsou nastaveny parametry v trvani cele simulace **/
      FIN_MT (tmm_closure_init ());

  /* Definice zapisu logu pro vsechny zpravy tykajici se inicializace   */
      tmm_problem_log_handle = op_prg_log_handle_create (
            OpC_Log_Category_Configuration,
            "TMM", "closure stage loading of propagation model", 20);

  /* Jestlize jsou uzly mimo platnou vypocetni oblast,     */
  /* hrozi generovani prilis velkeho poctu zaznamu do logu.      */
  /* Zastaveni zaznamu pri poctu vyssim nez 500  */

      DraS_TMM_Verbose_Log_H = op_prg_log_handle_create (
            OpC_Log_Category_Lowlevel,
            "TMM", "path loss calculation", 500);

  /* Zjistei, zda jsme v rezimu, ktery vyzaduje TMM */
      if (prg_env_attr_value_get (PrgC_Env_Attr_Boolean, TMMC_ENV_SIMULATE,
                  &using_tmm) == PrgC_Compcode_Failure)
            using_tmm = OPC_FALSE;

  /*  Overeni, zda je TMM aktivni      */
      if (using_tmm == OPC_FALSE)
            {

    /* Simulace pro vypocet ztraty sireni a dostupnosti nevyuziva TMM   */
    /* Nastaveni ne TMM metody, ktera bude uzita */
            dra_non_tmm_closure_method_set (OP_SIM_CONTEXT_PTR_OPT);
            }
      else
            {

    /* Simulace vyuziva TMM*/

    /* Nastaveni priznaku pro kontrolu uspesneho nacteni modulu TMM*/
            load_successful = OPC_FALSE;
```

```
    /* Pokus o naceni vychoziho TMM modelu sireni     */
             DraS_Closure_Tmm_Prop_Model_Name = tmm_default_propagation_model_get ();
             if (strcmp ("NONE", DraS_Closure_Tmm_Prop_Model_Name) == 0)
                    {

      /* Jestlize model sireni ma nazev NONE, jde o specialni pripad a  */
      /* indikuje, ze nebude nastaven zadny model sireni. */
                    load_successful = OPC_FALSE;

      /* Simulace nevyuziva TMM. Bude vyuzita jina metoda sireni  */
                    dra_non_tmm_closure_method_set (OP_SIM_CONTEXT_PTR_OPT);

                    FOUT
                    }

    /* Nacteni TMM modelu sireni */
             DraS_Closure_Prop_Model_Ptr = tmm_propagation_model_get
(DraS_Closure_Tmm_Prop_Model_Name);

    /* Kontrola spravneho nacteni TMM modelu sireni  */
             if (DraS_Closure_Prop_Model_Ptr == OPC_NIL)
                    {

      /* Model se nepodarilo uspesne nacist. Pravdepdobne chyby v mod_dirs     */
                    load_successful = OPC_FALSE;
      /* Priprava zpravy pro simulaci.                                         */
                    sprintf (line0_buf, "TMM: unable to load propagation model (%s)",
                           DraS_Closure_Tmm_Prop_Model_Name);
                    strcpy (line1_buf, "Using default closure instead.");

      /* Tisk zpravy                                                    */
                    op_sim_message (line0_buf, line1_buf);

      /* Zapis zpravy do logu*/
                    op_prg_log_entry_write (tmm_problem_log_handle,
                           "Pipeline stage 'closure' during initialization for\n"
                           "%s\n"
                           "%s\n"
                           "\n"
                           "Check that your mod_dirs contains all 3 of the needed\n"
                           "propagation model files (.prop.d, .prop.p and
.prop.[so/dll]).\n",
                           line0_buf,
                           line1_buf);
                    }
             else if (DraS_Closure_Prop_Model_Ptr->initialized_ok_flag == OPC_FALSE)
                    {

       /* Inicializacni funkce modelu sireni nastavi priznak      */
      /* "initialized_ok_flag" pro indikaci potizi uvnitr modelu */
                    load_successful = OPC_FALSE;

      /* Priprava zpravy pro simulaci  */
                    sprintf (line0_buf,
                           "TMM propagation model (%s) reported an initialization
problem.",
                           DraS_Closure_Tmm_Prop_Model_Name);
                    strcpy (line1_buf, "Using default closure instead.");

      /* Tisk zpravy */
                    op_sim_message (line0_buf, line1_buf);

      /* Zapis zpravy do logu*/
                    op_prg_log_entry_write (tmm_problem_log_handle,
                           "Pipeline stage 'closure' during initialization for TMM:\n"
                           "%s\n"
                           "%s\n",
                           line0_buf,
```

```
                        line1_buf);
                }
            else
                {

      /* Model sireni byl uspesne nacten      */
      /* Ztraty sireni budeme pocitat z nacteneho modelu   */
                load_successful = OPC_TRUE;

      /* Indikace uziti TMM modelu */
                DraS_Active_Closure_Method = DraC_Terrain_Modeling;

      /* Jestlize je nastaven vypis zprav, zapis zpravu do logu   */
                if (tmm_verbose_get ())
                        {
                        sprintf (line0_buf,
                                "TMM initialization: successfully loaded the propagation
model (%s)",
                                DraS_Closure_Tmm_Prop_Model_Name);
                        op_prg_log_entry_write (tmm_problem_log_handle,
                                line0_buf);
                        }
                }

    /*      Kontrola, zda TMM modul byl uspesne nacten*/
            if (load_successful == OPC_FALSE)
                {

      /* Simulace nebude vyuzivat pro vypocetztrat a dostupnosti TMM model */
      /* Bude uzita ne TMM metoda dostupnosti       */
                dra_non_tmm_closure_method_set (OP_SIM_CONTEXT_PTR_OPT);
                }
            }
       FOUT
       }

static void
dra_non_tmm_closure_method_set (OP_SIM_CONTEXT_ARG_OPT)
       {
       int                      attr_value;

  /** Tato funkce je volana pro ziskani metody, kter bude pouzita **/
  /** namisto TMM. K dispozici jsou dve moznosti **/
  /** - Prenosova cesta bude vzdy ddstupna **/
  /** - Line-of-sight s kulovym modelem Zeme**/
  /** **/
  /** Jsou dve moznosti, kterou ne TMM metodu zvolit **/
  /** **/
  /** A) Pro standardni modely bezdratovych siti: Metoda pro urceni**/
  /** dostupnosti spojeni je volena podle nastaveni globalniho atributu**/
  /** CLOSURE_METHOD_GLOBAL_ATTRIBUTE_NAME **/
  /** **/
  /** B) Pro ostatni modely bezdratovych siti: Jestlize globalni atribut **/
  /** neni definovan, tak bude vzdy pouzita metoda Earth Line-of-sight **/
       FIN_MT(dra_non_tmm_closure_method_set (void));

  /* Kontrola, jestlize globalni atribut je definovan      */
  /* Jestlize ano, zvol postup A, jinak to znamena, ze     */
  /* bude simulovan uzivatelska bezdratova sit B */
       if (op_ima_sim_attr_exists (CLOSURE_METHOD_GLOBAL_ATTRIBUTE_NAME))
            {
            if (op_ima_sim_attr_get_int32 (CLOSURE_METHOD_GLOBAL_ATTRIBUTE_NAME,
&attr_value) == OPC_COMPCODE_SUCCESS)
                {

      /* Nastaveni metody podle globalniho atributu */
                DraS_Active_Closure_Method = (DraT_Closure_Method) attr_value;
                }
```

```
            else
                  {

      /* U uzivatelskych siti bude pouzit vzdy Line-of-Sight metoda     */
                  DraS_Active_Closure_Method = DraC_Line_Of_Sight_Never_Occluded;
                  }
            }
      else
            {

    /* U uzivatelskych siti bude pouzit vzdy Line-of-Sight metoda */
            DraS_Active_Closure_Method = DraC_Earth_Line_Of_Sight;
            }
      FOUT;
      }

static void
tmm_model_closure_calc (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
      {
      TmmT_Positiontx_position;
      TmmT_Positionrx_position;
      void *              pipeline_invocation_state_ptr;
      double              tx_base_freq;
      double              tx_bandwidth;
      double              tx_center_freq;
      int                       verbose_active;
      int                       trace_active;
      int                       str_index;
      char *              msg_str_ptrs [TMMC_LOSS_MESSAGE_BUF_NUM_STRS];
      char                log_str_buf [16 * TMMC_LOSS_MESSAGE_BUF_STR_SIZE];
      double              tmm_model_path_loss_dB;
      TmmT_Loss_Status    tmm_model_loss_status;
      char                tmm_model_msg_buffer_v [TMMC_LOSS_MESSAGE_BUF_NUM_STRS]
[TMMC_LOSS_MESSAGE_BUF_STR_SIZE];
      char                msg_buf0 [256];
      char                msg_buf1 [256];
      char                msg_buf2 [256];
      char                msg_buf3 [256];
      char                msg_buf4 [256];

  /** Volani modelu sireni signalu. Funkce muze informovat o  **/
  /** - utlumu vlivem sireni signalu **/
  /** - chybovem stavu **/
  /** - nedostupnosti prijimaciho kanalu, komunikace neni mozna **/
      FIN_MT (tmm_model_closure_calc (pkptr));

    /* Nacteni prenosove frekvence v Hz. */
      tx_base_freq = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_FREQ);
      tx_bandwidth = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_BW);
      tx_center_freq = tx_base_freq + (tx_bandwidth / 2.0);

  /* Nacteni pozice vysilace. */
      tx_position.latitude = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_LAT);
      tx_position.longitude = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_LONG);
      tx_position.elevation = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_ALT);

  /* Nacteni pozice prijimace. */
      rx_position.latitude = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_LAT);
      rx_position.longitude = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_LONG);
      rx_position.elevation = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_ALT);

  /* Modely sireni dodavane OM neuzivaji zadne extra stavy */
  /* Pro prehlednost je nastaven lokalni ukazatel stavu na nulu */
  /* Uzivatelem definovane modely mohou zavest jine libovolne hodnoty */
      pipeline_invocation_state_ptr = OPC_NIL;

  /* Zjisteni, zda je nastaveno informovani o stavu TMM. */
  /* Muze byt nastaveno pomoci tmm_verbose atributu. */
```

```
      trace_active = op_prg_odb_trace_active ()||op_prg_odb_pktrace_active (pkptr);
      verbose_active = tmm_verbose_get ();

  /*** Volani vypoctu utlumu signalu                ***/

      /* Vsechny TMM model maji prototypovou funkci utlumu sigmalu nasledujici*/
      /* z tmm.h*/
      /*
            double TmmT_Path_Loss_Calc_Method (
                        TmmT_Propagation_Model *  model_ptr,
                        void *                                  tda_state_ptr,
                        TmmT_Position *                 tx_position_ptr,
                        TmmT_Position *                 rx_position_ptr,
                        double                                  center_frequency,
                        double                                  bandwidth,
                        int                                     verbose,
                        TmmT_Loss_Status *              status_ptr,
                        char                                    message_buffer_v
[TMMC_LOSS_MESSAGE_BUF_NUM_STRS] [TMMC_LOSS_MESSAGE_BUF_STR_SIZE]);
      */
      for (str_index = 0; str_index < TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
            tmm_model_msg_buffer_v [str_index] [0] = '\0';
      tmm_model_path_loss_dB =
            DraS_Closure_Prop_Model_Ptr->path_loss_calc_method (
                  DraS_Closure_Prop_Model_Ptr,
                  pipeline_invocation_state_ptr,
                  &tx_position,
                  &rx_position,
                  tx_center_freq,
                  tx_bandwidth,
                  verbose_active || trace_active,
                  &tmm_model_loss_status,
                  tmm_model_msg_buffer_v );

      if (tmm_model_loss_status == TmmC_Loss_No_Closure)
            {

    /* Model sireni hlasi, ze signal tohoto paketu nebude  */
    /* prijimacem dosazitelny. K Doruceni tohoto paketu nedojde   */
            if (trace_active || verbose_active)
                  {
                  sprintf (msg_buf0, "Transmission Closure: Propagation Model %s:",
                        DraS_Closure_Tmm_Prop_Model_Name );
                  sprintf (msg_buf1, "Path loss between transmitter (ID %d) and
receiver (ID %d)",
                        op_td_get_int (pkptr, OPC_TDA_RA_TX_OBJID),
                        op_td_get_int (pkptr, OPC_TDA_RA_RX_OBJID));
                  strcpy (msg_buf2, "fails closure. Packet can not be received by the
receiver");
                  strcpy (msg_buf3, "Additional messages reported by propagation model
follow:");
                  for (str_index = 0; str_index < TMMC_LOSS_MESSAGE_BUF_NUM_STRS;
str_index++)
                        {
                        if (tmm_model_msg_buffer_v [str_index] [0] == '\0')
                              {

          /* Nulovy ukazatel na op_prg_odb_print_ znaci posledni argument */
                              msg_str_ptrs [str_index] = OPC_NIL;
                              break;
                              }
                        else
                              {
                              msg_str_ptrs [str_index] = &tmm_model_msg_buffer_v
[str_index] [0];
                              }
                        }
                  if (msg_str_ptrs [0] == OPC_NIL)
```

```
                        {
                        strcpy (msg_buf3, "<no messages reported by the propagation
model>");
                        }
                  }
            if (trace_active)
                  {

      /* Tisk zpravy do ODB */
                  op_prg_odb_print_major (msg_buf0, msg_buf1, msg_buf2, msg_buf3,
                        msg_str_ptrs [0], msg_str_ptrs [1], msg_str_ptrs [2],
msg_str_ptrs [3],
                        msg_str_ptrs [4], OPC_NIL);
                  }
            if (verbose_active)
                  {

      /* Zacit logovat simulaci protoze tmm_verbose je nastaen    */
      /* Zaznamy v logu jsou predavany pomoci 1 dlouheho retezce  */

                  log_str_buf [0] = '\0';
                  strcat (log_str_buf, msg_buf0);
                  strcat (log_str_buf, "\n");

                  strcat (log_str_buf, "  ");
                  strcat (log_str_buf, msg_buf1);
                  strcat (log_str_buf, "\n");

                  strcat (log_str_buf, "  ");
                  strcat (log_str_buf, msg_buf2);
                  strcat (log_str_buf, "\n");

                  strcat (log_str_buf, "  ");
                  strcat (log_str_buf, msg_buf3);
                  strcat (log_str_buf, "\n");

                  for (str_index = 0; str_index < TMMC_LOSS_MESSAGE_BUF_NUM_STRS;
str_index++)
                        {
                        if (tmm_model_msg_buffer_v [str_index] [0] != '\0')
                              {
                              strcat (log_str_buf, "    ");
                              strcat (log_str_buf, tmm_model_msg_buffer_v
[str_index]);
                              strcat (log_str_buf, "\n");
                              }
                        else
                              {

          /* Prazny retezec, takze nejsou dalsi zpravy     */
                              break;
                              }
                        }

      /* Loguj zpravu */
                  op_prg_log_entry_write (DraS_TMM_Verbose_Log_H,
                        log_str_buf);
                  }
            op_td_set_int (pkptr, OPC_TDA_RA_CLOSURE, OPC_FALSE);
            }
       else if (tmm_model_loss_status == TmmC_Loss_Error)
            {
    /* V modelu sireni byl zaznamenan chybovy stav   */
    /* Bude volan simple-earth model sireni   */
            sprintf (msg_buf0, "Transmission Closure: Propagation Model %s:",
                  DraS_Closure_Tmm_Prop_Model_Name );
            sprintf (msg_buf1, "Model reported error in computing path between
transmitter (ID %d) and receiver (ID %d)",
```

```
                op_td_get_int (pkptr, OPC_TDA_RA_TX_OBJID),
                op_td_get_int (pkptr, OPC_TDA_RA_RX_OBJID));
          strcpy (msg_buf2, "Using simple-earth closure model for this transmission");
          strcpy (msg_buf3, "Messages reported by propagation model follow:");
          for (str_index = 0; str_index < TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
                {
                if (tmm_model_msg_buffer_v [str_index] [0] == '\0')
                      {

      /* Nulovy ukazatel op_prg_odb_print_ bude ukazovat posledni argument */
                      msg_str_ptrs [str_index] = OPC_NIL;
                      break;
                      }
                else
                      {
                      msg_str_ptrs [str_index] = &tmm_model_msg_buffer_v [str_index]
[0];
                      }
                }
          if (msg_str_ptrs [0] == OPC_NIL)
                {
                strcpy (msg_buf3, "<no messages reported by the propagation model>");
                }
          if (trace_active)
                {

    /* Tisni zpravu do ODB */
                op_prg_odb_print_major (msg_buf0, msg_buf1, msg_buf2, msg_buf3,
                      msg_str_ptrs [0], msg_str_ptrs [1], msg_str_ptrs [2],
msg_str_ptrs [3],
                      msg_str_ptrs [4], OPC_NIL);
                }
    /* Zaznamy v logu jsou predavany pomoci 1 dlouheho retezce    */
          log_str_buf [0] = '\0';
          strcat (log_str_buf, msg_buf0);
          strcat (log_str_buf, "\n");

          strcat (log_str_buf, "  ");
          strcat (log_str_buf, msg_buf1);
          strcat (log_str_buf, "\n");

          strcat (log_str_buf, "  ");
          strcat (log_str_buf, msg_buf2);
          strcat (log_str_buf, "\n");

          strcat (log_str_buf, "  ");
          strcat (log_str_buf, msg_buf3);
          strcat (log_str_buf, "\n");

          for (str_index = 0; str_index < TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
                {
                if (tmm_model_msg_buffer_v [str_index] [0] != '\0')
                      {
                      strcat (log_str_buf, "    ");
                      strcat (log_str_buf, tmm_model_msg_buffer_v [str_index]);
                      strcat (log_str_buf, "\n");
                      }
                else
                      {

      /* Prazny retezec, takze nejsou dalsi zpravy */
                      break;
                      }
                }

    /* Zprava logu indikuje, ze u paketu doslo ke zmene modelu sireni  */
    /* na freespace */
          op_prg_log_entry_write (DraS_TMM_Verbose_Log_H,
```

```
                log_str_buf);

    /* Spusten zalozni jednoduchy Line-of-sight model             */
            generic_earth_LOS_closure (OP_SIM_CONTEXT_PTR_OPT_COMMA pkptr);
            }
       else
            {

    /* Model sireni bude schopen vpocitat hodnotu signalu. */
            if (tmm_model_path_loss_dB > LOSS_CUTOFF_THRESHOLD_DB)
                  {

      /* Signal tohoto paketu je dostatecne silny.  Uroven signalu*/
      /* je vyssi nez mezni hodnota. Dfinovano v LOSS_CUTOFF_THRESHOLD_DB)*/

      /* Ztrata sirenim je v mezich pro prijem. Mame navazane spojeni */
                  op_td_set_int (pkptr, OPC_TDA_RA_CLOSURE, OPC_TRUE);

       /* Kroky prenosu pro silu signalu budou pocitat konecou silu prijateho */
      /* signalu (vcetne zisku anteny, vykonu vysilace a pod.) */
      /* Nyni nastavime utlum sirenim v prislusnem TDA */

      /* The default power pipeline stage will use a convention of      */
       /* interpreting this path loss in raw form (not in dB).    */

      /* Vychozi model kroku sily prijateho signalu nebude pouzivat   */
      /* pro definovani ztrat sirenim hodnoty v dB */

                  op_td_set_dbl (pkptr, OPC_TDA_RA_RCVD_POWER,
                        pow (10.0, tmm_model_path_loss_dB / 10.0));

      /* Jestlize je povoleno, zapise zpravu do ODB  */
                  if (trace_active)
                        {
                        op_prg_odb_print_major ("Successful path loss computation
reported by TMM.", OPC_NIL);

        /* Pripjit taky dalsi pripadne zpravy */
                        for (str_index = 0; str_index <
TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
                              {
                              if (tmm_model_msg_buffer_v [str_index][0] != '\0')
                                    {
                                    op_prg_odb_print_minor (tmm_model_msg_buffer_v
[str_index], OPC_NIL);
                                    }
                              else
                                    {

            /* Prazdny retezec indikuje konec zprav. */
            /* Ukonceni smcky.   */
                                    break;
                                    }
                              }
                        }
                  }
             else
                  {
      /* Ztraty sirenim jsu prilis velke, takze prijmany signal bude    */
      /* prilis maly. Paket nedosahne pozadovaneho spojeni  */
      /* Mezni hranice hodnoty urovne signalu je definovana na zacatku */
      /* souboru. Uzitim teto mezni hodnoty můžeme vyrazne zkvalitnit   */
      /* simulaci uzitim mensiho poctu kroku prenosu  */
                  if (trace_active || verbose_active)
                        {
                        sprintf (msg_buf0, "Transmission Closure: Propagation Model
%s:",
                              DraS_Closure_Tmm_Prop_Model_Name );
```

```
                        sprintf (msg_buf1, "Path loss between transmitter (ID %d) and
receiver (ID %d)",
                                op_td_get_int (pkptr, OPC_TDA_RA_TX_OBJID),
                                op_td_get_int (pkptr, OPC_TDA_RA_RX_OBJID));
                        sprintf (msg_buf2, "is beyond threshold of %f dB.",
                                LOSS_CUTOFF_THRESHOLD_DB);
                        strcpy (msg_buf3, "Transmission is considered to fail
closure");
                        strcpy (msg_buf4, "Additional messages reported by propagation
model follow:");
                        for (str_index = 0; str_index <
TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
                                {
                                if (tmm_model_msg_buffer_v [str_index] [0] == '\0')
                                        {

            /* Nulovy ukazatel op_prg_odb_print_ bude ukazovat posledni argument */
                                        msg_str_ptrs [str_index] = OPC_NIL;
                                        break;
                                        }
                                else
                                        {
                                        msg_str_ptrs [str_index] = &
tmm_model_msg_buffer_v [str_index] [0];
                                        }
                                }
                        if (msg_str_ptrs [0] == OPC_NIL)
                                {
                                strcpy (msg_buf4, "<no messages reported by the
propagation model>");
                                }
                        }
                    if (trace_active)
                        {

        /* Tisk zpravy do ODB */
                        op_prg_odb_print_major (msg_buf0, msg_buf1, msg_buf2,
msg_buf3, msg_buf4,
                                msg_str_ptrs [0], msg_str_ptrs [1], msg_str_ptrs [2],
msg_str_ptrs [3],
                                msg_str_ptrs [4], OPC_NIL);
                        }
                    if (verbose_active)
                        {

        /* Zacit logovat simulaci protoze tmm_verbose je nastaen  */

        /* Zaznamy v logu jsou predavany pomoci 1 dlouheho retezce      */
                        log_str_buf [0] = '\0';
                        strcat (log_str_buf, msg_buf0);
                        strcat (log_str_buf, "\n");

                        strcat (log_str_buf, "  ");
                        strcat (log_str_buf, msg_buf1);
                        strcat (log_str_buf, "\n");

                        strcat (log_str_buf, "  ");
                        strcat (log_str_buf, msg_buf2);
                        strcat (log_str_buf, "\n");

                        strcat (log_str_buf, "  ");
                        strcat (log_str_buf, msg_buf3);
                        strcat (log_str_buf, "\n");

                        strcat (log_str_buf, "  ");
                        strcat (log_str_buf, msg_buf4);
                        strcat (log_str_buf, "\n");
```

```
                        for (str_index = 0; str_index <
TMMC_LOSS_MESSAGE_BUF_NUM_STRS; str_index++)
                                {
                                if (tmm_model_msg_buffer_v [str_index] [0] != '\0')
                                        {
                                        strcat (log_str_buf, "    ");
                                        strcat (log_str_buf, tmm_model_msg_buffer_v
[str_index]);
                                        strcat (log_str_buf, "\n");
                                        }
                                else
                                        {
                                        /* Empty string, so no more string in message
buffer from propagation model    */
            /* Prazny retezec, takze nejsou dalsi zpravy   */
                                        break;
                                        }
                                }

             /* Zapis zpravu */
                                op_prg_log_entry_write (DraS_TMM_Verbose_Log_H,
                                        log_str_buf);
                                }
                        op_td_set_int (pkptr, OPC_TDA_RA_CLOSURE, OPC_FALSE);
                        }
             }
      FOUT
      }
```

PŘÍLOHA E Kód modelu 3. kroku dra_chanmatch

```
/* dra_chanmatch.ps.c */
/* Vychozi model vyberu kanalu radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009       */
/*           by OPNET Technologies, Inc.             */
/*           (A Delaware Corporation)        */
/*     7255 Woodmont Av., Suite 250          */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.             */
/*****************************************/

#include "opnet.h"

#if defined (__cplusplus)
extern "C"
#endif
void
dra_chanmatch_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
      {
      double       tx_freq, tx_bw, tx_drate, tx_code;
      double       rx_freq, rx_bw, rx_drate, rx_code;
      Vartype            tx_mod;
      Vartype            rx_mod;

      /** Stanoveni kompatibility mezi vysilacim a prijimacim kanalem. **/
  FIN_MT (dra_chanmatch (pkptr));

  /* Nacteni parametru vysilaciho kanalu. */
      tx_freq            = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_FREQ);
      tx_bw        = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_BW);
      tx_drate     = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_DRATE);
      tx_code            = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_CODE);
      tx_mod       = op_td_get_ptr (pkptr, OPC_TDA_RA_TX_MOD);

  /* Nacteni parametru prijimaciho kanalu. */
      rx_freq            = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_FREQ);
      rx_bw        = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_BW);
      rx_drate     = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_DRATE);
      rx_code            = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_CODE);
      rx_mod       = op_td_get_ptr (pkptr, OPC_TDA_RA_RX_MOD);

  /* Pri neprekryvajicim se pasmu paket nema na prijimaci kanal vliv.   */
  /* Tyto pakety jsou zcela ignorovany.*/

      if ((tx_freq > rx_freq + rx_bw) || (tx_freq + tx_bw < rx_freq))
            {
            op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS, OPC_TDA_RA_MATCH_IGNORE);
            FOUT
            }

  /* Pokud paket projde, testujeme rozdily atributu kanalu v pasmu,   */
  /* ktere mohou rozhodnut o zarazeni paketu do sumu  */
      if ((tx_freq != rx_freq) || (tx_bw != rx_bw) ||
            (tx_drate != rx_drate) || (tx_code != rx_code) || (tx_mod != rx_mod))
            {
            op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS, OPC_TDA_RA_MATCH_NOISE);
            FOUT
            }

  /* Pokud paket projde a k tomuto mistu, paket je oznacen za platny */
  /* a pripadne prijaty v kroku korekce chyb  */
      op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS, OPC_TDA_RA_MATCH_VALID);
      FOUT
      }
```

PŘÍLOHA F Kód modelu 4.kroku dra_tagain

```
/* dra_tagain.ps.c */
/* Vychozi model kroku "zisk vysilaci anteny" radioveho prenosu */

/*****************************************/
/*       Copyright (c) 1993-2009         */
/*      by OPNET Technologies, Inc.      */
/*       (A Delaware Corporation)        */
/*    7255 Woodmont Av., Suite 250       */
/*     Bethesda, MD 20814, U.S.A.        */
/*         All Rights Reserved.          */
/*****************************************/

#include "opnet.h"
#include <math.h>

#if defined (__cplusplus)
extern "C"
#endif
void
dra_tagain_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
    {
      double       rx_x, rx_y, rx_z;
      double       gain;
      Vartype            pattern_table;
      Objid        tx_antenna_objid;
      double             azimuth, polar;

  /** Vypocet zisku souvidejiciho s vysilaci antenou.      **/
      FIN_MT (dra_tagain (pkptr));

  /* Nacteni dat pro zpracovani zisku vysilaci anteny. */
      pattern_table = op_td_get_ptr (pkptr, OPC_TDA_RA_TX_PATTERN);

  /* Specialni pripad: podle zvyklosti ukazatel s nulovu hodntou ukazuje */
  /* na adresu s daty pro vsesmerovou antenu. Nejsou nutne zadne vypocty*/
      if (pattern_table == OPC_NIL)
            {
            /* Assign zero dB gain regardless of transmission direction. */
    /* Prirad 0dB bez ohledu na smer prenosu. */
            op_td_set_dbl (pkptr, OPC_TDA_RA_TX_GAIN, 0.0);
            FOUT
            }

  /* Nacti geocentricke souradnice prijimace. */
      rx_x = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_GEO_X);
      rx_y = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_GEO_Y);
      rx_z = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_GEO_Z);

  /* Nacti OBJID vysilaci anteny . */
      tx_antenna_objid = op_td_get_int (pkptr, OPC_TDA_RA_TX_ANT_OBJID);

      if (OPC_COMPCODE_SUCCESS == op_radio_antenna_angles_to_location_get
(tx_antenna_objid, rx_x, rx_y, rx_z,
            &polar, &azimuth))
            {
            gain = op_tbl_pat_gain(pattern_table, polar, azimuth);
            }
      else
            {
            gain = 0.0;
            }
#ifndef OPD_NO_DEBUG
      if (op_prg_odb_ltrace_active ("antenna_gain"))
            {
            Objid  rx_antenna_objid;
```

```
            char test_str [256];

            rx_antenna_objid = op_td_get_int (pkptr, OPC_TDA_RA_RX_ANT_OBJID);
            if ((azimuth!= OPC_DBL_INVALID) && (polar!=OPC_DBL_INVALID))
                  {
                  sprintf (test_str, "\n TX gain at angles (azimuth = %.3f, polar =
%.3f) is %.3f\n", azimuth, polar, gain);
                  }
            else
                  {

      /* V pripade vsesmerove anteny nejsou azimut a polarni souradnice uvazovany. */
                  sprintf (test_str, "\n TX gain for isotropic antenna is %.3f\n",
gain);
                  }

            op_prg_odb_print_major (test_str, OPC_NIL);
            sprintf (test_str, " for TX antenna objid (" OPC_OBJ_ID_FMT "), Rx antenna
objid (" OPC_OBJ_ID_FMT ").\n",
                  tx_antenna_objid,
                  rx_antenna_objid);
            op_prg_odb_print_minor (test_str, OPC_NIL);
            }
#endif

  /* Uloz zisk vysilaci anteny do prislusneho TDA. */
      op_td_set_dbl (pkptr, OPC_TDA_RA_TX_GAIN, gain);
      FOUT;
      }
```

PŘÍLOHA G Kód modelu 5. kroku dra_propdel

```
/* dra_propdel.ps.c */
/* Vychozi model kroku zpozdeni vysilani radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009       */
/*           by OPNET Technologies, Inc.             */
/*           (A Delaware Corporation)        */
/*     7255 Woodmont Av., Suite 250          */
/*     Bethesda, MD 20814, U.S.A.       */
/*                 All Rights Reserved.            */
/*****************************************/

#include "opnet.h"

/***** konstanty *****/

/* rychlost sireni radioveho signalu (m/s) */
#define      PROP_VELOCITY3.0E+08

/***** Procedura prenosu *****/

#if defined (__cplusplus)
extern "C"
#endif
void
dra_propdel_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
      {
      double       start_prop_delay, end_prop_delay;
      double       start_prop_distance, end_prop_distance;

  /** Vypocet zpozdeni prenosu mezi vysilacem a prijimacem **/

      FIN_MT (dra_propdel (pkptr));

  /* Nacteni vychozi vdalenosti mezi vysilacem a prijimacem */
      start_prop_distance = op_td_get_dbl (pkptr, OPC_TDA_RA_START_DIST);

  /* Nacteni konecne vdalenosti mezi vysilacem a prijimacem */
      end_prop_distance = op_td_get_dbl (pkptr, OPC_TDA_RA_END_DIST);

  /* Vypocet zpozdeni sireni signalu na zacatku prijmu */
      start_prop_delay = start_prop_distance / PROP_VELOCITY;

  /* Vypocet zpozdeni sireni signalu na konci prijmu */
      end_prop_delay = end_prop_distance / PROP_VELOCITY;

  /* Ulozeni obou zpozdeni prenosu signalu do prislusnych TDA */
      op_td_set_dbl (pkptr, OPC_TDA_RA_START_PROPDEL, start_prop_delay);
      op_td_set_dbl (pkptr, OPC_TDA_RA_END_PROPDEL, end_prop_delay);

      FOUT
      }
```

PŘÍLOHA H Kód modelu 6. kroku dra_ragain

```
/* dra_ragain.ps.c */
/* Vychozi model kroku "zisk prijimaci anteny" radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009       */
/*           by OPNET Technologies, Inc.             */
/*           (A Delaware Corporation)        */
/*     7255 Woodmont Av., Suite 250          */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.             */
/*****************************************/

#include "opnet.h"
#include <math.h>

#if defined (__cplusplus)
extern "C"
#endif
void
dra_ragain_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       double       tx_x, tx_y, tx_z;
       double              gain;
       Vartype             pattern_table;
       Objid        rx_antenna_objid;
       double       azimuth, polar;

  /** Vypocet zisku souvidejiciho s prijimaci antenou.     **/
       FIN_MT (dra_ragain (pkptr));

  /* Nacteni dat pro zpracovani zisku prijimaci anteny. */
       pattern_table = op_td_get_ptr (pkptr, OPC_TDA_RA_RX_PATTERN);

  /* Specialni pripad: podle zvyklosti ukazatel s nulovu hodntou ukazuje */
  /* na adresu s daty pro vsesmerovou antenu. Nejsou nutne zadne vypocty*/
       if (pattern_table == OPC_NIL)
              {

    /* Prirad 0dB bez ohledu na smer prenosu. */
              op_td_set_dbl (pkptr, OPC_TDA_RA_RX_GAIN, 0.0);
              FOUT
              }

  /* Nacti geocentricke souradnice vysilace. */
       tx_x = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_GEO_X);
       tx_y = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_GEO_Y);
       tx_z = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_GEO_Z);

  /* Nacti OBJID prijmaci anteny . */
       rx_antenna_objid = op_td_get_int (pkptr, OPC_TDA_RA_RX_ANT_OBJID);

       if (OPC_COMPCODE_SUCCESS == op_radio_antenna_angles_to_location_get
(rx_antenna_objid,
              tx_x, tx_y, tx_z, &polar, &azimuth))
              {
              gain = op_tbl_pat_gain (pattern_table, polar, azimuth);
              }
       else
              {
              gain = 0.0;
              }

#ifndef OPD_NO_DEBUG
       if (op_prg_odb_ltrace_active ("antenna_gain"))
              {
              char test_str [256];
```

```
            if ((azimuth!= OPC_DBL_INVALID) && (polar!=OPC_DBL_INVALID))
                  {
                  sprintf (test_str, "\n RX gain at angles (azimuth = %.3f, polar =
%.3f) is %.3f\n", azimuth, polar, gain);
                  }
            else
                  {
                  sprintf (test_str, "\n RX gain at isotropic angles is %.3f\n", gain);
                  }

            op_prg_odb_print_major (test_str, OPC_NIL);
            sprintf (test_str, " for antenna objid (" OPC_OBJ_ID_FMT ").\n", (int)
rx_antenna_objid);
            op_prg_odb_print_minor (test_str, OPC_NIL);
            }
#endif

  /* Uloz zisk prijmaci anteny do prislusneho TDA. */
      op_td_set_dbl (pkptr, OPC_TDA_RA_RX_GAIN, gain);

      FOUT;
      }
```

PŘÍLOHA I Kód modelu 7. kroku dra_power

```
/* dra_power.ps.c */
/* Vychozi model kroku "sila prijateho signalu" radioveho prenosu*/
/* Tento model vyuziva stavove informace prijimaciho kanalu       */
/* pro kontrolu a aktualizaci stavu dostupnosti kanalu     */
/* To zavisi na vytvoreni a inicializaci stavovych informaci      */
/* kanalu v modelu rxgroup       */

/***************************************/
/*            Copyright (c) 1986-2009     */
/*           by OPNET Technologies, Inc    */
/*            (A Delaware Corporation)     */
/*        7255 Woodmont Av., Suite 250     */
/*      Bethesda, MD 20814, U.S.A.       */
/*               All Rights Reserved.       */
/***************************************/

#include "opnet.h"
#include "dra.h"
#include <math.h>

/***** konstanty *****/

#define C                               299792458.0
#define SIXTEEN_PI_SQ                   (16.0 * VOSC_NA_PI * VOSC_NA_PI)

/***** Procedura prenosu *****/

#if defined (__cplusplus)
extern "C"
#endif

void
dra_power_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
      {
      double       prop_distance, rcvd_power, path_loss;
      double       tx_power, tx_base_freq, tx_bandwidth, tx_center_freq;
      double       lambda, rx_ant_gain, tx_ant_gain;
      Objid              rx_ch_obid;
      double       in_band_tx_power, band_max, band_min;
      double       rx_base_freq, rx_bandwidth;
      DraT_Rxch_State_Info*     rxch_state_ptr;

  /** Vypocet prumerneho vykonu ve Wattech signalu   **/
  /** spojeneho s prenasenym paketem.   **/

      FIN_MT (dra_power (pkptr));

  /*Jestlize je prichazejici paket platny, muze jej prijimac zablokovat.*/
  /*Nicmene, pokud je prijimaci uzel odpojen, model vyberu kanalu jej */
  /* oznaci jako sum. */

      if (op_td_get_int (pkptr, OPC_TDA_RA_MATCH_STATUS) == OPC_TDA_RA_MATCH_VALID)
            {
            if (op_td_is_set (pkptr, OPC_TDA_RA_ND_FAIL))
                  {

      /* Prijimaci uzel je odpojen.  */
      /* Zmeni typ paketu na sum */
            op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS, OPC_TDA_RA_MATCH_NOISE);
                  }
            else
                  {

      /* Prijimaci uzel je pripraven.  */
      /* Nacti adresu prijimaciho kanalu */
            rx_ch_obid = op_td_get_int (pkptr, OPC_TDA_RA_RX_CH_OBJID);
```

```
      /* Pristup k stavovym informacim prijimaciho kanalu */
             rxch_state_ptr = (DraT_Rxch_State_Info *) op_ima_obj_state_get (rx_ch_obid);

      /* Je-li kanal prijimace jiz uzamcen, paket bude nyni  */
      /* povazovan za sum. To brani soucasnemu prijmu  */
      /* vice platnych paketu v jednom kanalu. */

                    if (rxch_state_ptr->signal_lock)
                          op_td_set_int (pkptr, OPC_TDA_RA_MATCH_STATUS,
OPC_TDA_RA_MATCH_NOISE);
                    else
                          {

        /* V opacnem pripade bude kanal prijimace uzamcen, */
        /* dokud nebude prijem paketu zcela dokoncen */
                          rxch_state_ptr->signal_lock = OPC_TRUE;
                          }
                    }
             }

  /* Nacteni prideleho vykonu vysilaciho kanalu  */
       tx_power = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_POWER);

  /* Nacteni prenosove frekvence v Hz */
       tx_base_freq = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_FREQ);
       tx_bandwidth = op_td_get_dbl (pkptr, OPC_TDA_RA_TX_BW);
       tx_center_freq = tx_base_freq + (tx_bandwidth / 2.0);

  /* Vypocet vlnove delky v metrech */
       lambda = C / tx_center_freq;

  /* Nacteni vzdalenosti mezi vysilacem a prijimacem v metrech */
       prop_distance = op_td_get_dbl (pkptr, OPC_TDA_RA_START_DIST);

  /* Pokud je vyuzivano TMM, atribut TDA OPC_TDA_RA_RCVD_POWER  */
  /* ma jiz prvotni hodnotu ztrat sirenim vlvneni  */
       if (op_td_is_set (pkptr, OPC_TDA_RA_RCVD_POWER))
             {
             path_loss = op_td_get_dbl (pkptr, OPC_TDA_RA_RCVD_POWER);
             }
       else
             {

     /* Vypocet ztrat sirenim pro tuto vzdalenost a vlnovou delku */
             if (prop_distance > 0.0)
                    {
                    path_loss = (lambda * lambda) /
                          (SIXTEEN_PI_SQ * prop_distance * prop_distance);

       /* Model "Freespace" je platny jen pro skutecne rozlehle site, */
       /* kde vysledek skutecne zavisi na vlnove delce lambda a vykonem  */
       /* uzlu. Jelikoz nezname vykonove pomery, je treba se ujistit, ze   */
       /* lambda neprekroci hodnotu 1 */
                    if (path_loss > 1.0)
                          path_loss = 1.0;
                    }
             else
                    path_loss = 1.0;
             }

  /* Nacteni sirky pasma a zakladni frekvence */
       rx_base_freq = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_FREQ);
       rx_bandwidth = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_BW);

  /* Pouziti techto hodnot k urceni pasma prekryvajiciho vysilaci uzel. */
  /* Neni zde pripad zadneho prekryti pasem, protoze tyto pakety jsou jiz  */
  /* filtrovany v modelu vyberu kanalu */
```

```
/* Zakladni frekvenci pasma prekryti je vyssi zakladni frekvence  */
     if (rx_base_freq > tx_base_freq)
           band_min = rx_base_freq;
     else
           band_min = tx_base_freq;

/* Mezni horni kmitocet pasma prekryti je nejnizsi prenasena frekvence */
     if (rx_base_freq + rx_bandwidth > tx_base_freq + tx_bandwidth)
           band_max = tx_base_freq + tx_bandwidth;
     else
           band_max = rx_base_freq + rx_bandwidth;

/* Vypocet pasmoveho vykonu vysilace */
     in_band_tx_power = tx_power * (band_max - band_min) / tx_bandwidth;

/* Nacteni zisku anten (ne v dB) */
     tx_ant_gain = pow (10.0, op_td_get_dbl (pkptr, OPC_TDA_RA_TX_GAIN) / 10.0);
     rx_ant_gain = pow (10.0, op_td_get_dbl (pkptr, OPC_TDA_RA_RX_GAIN) / 10.0);

/* Vypocet prijimaneho vykonu */
     rcvd_power = in_band_tx_power * tx_ant_gain * path_loss * rx_ant_gain;

/* Nastaveni prijimaneho vykonu ve Wattech v prislusnem TDA  */
     op_td_set_dbl (pkptr, OPC_TDA_RA_RCVD_POWER, rcvd_power);

     FOUT
     }
```

PŘÍLOHA J Kód modelu 8. kroku dra_inoise

```
/* dra_inoise.ps.c */
/* Vychozi model kroku interferencni sum radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009       */
/*           by OPNET Technologies, Inc.            */
/*           (A Delaware Corporation)        */
/*     7255 Woodmont Av., Suite 250          */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.            */
/*****************************************/

#include "opnet.h"

#if defined (__cplusplus)
extern "C"
#endif
void
dra_inoise_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr_prev, Packet * pkptr_arriv)
	{
	int                arriv_match, prev_match;
	double       prev_rcvd_power, arriv_rcvd_power;

	/** Vhodnoceni kolize paketu behem prichodu 'pkptr_arriv', **/
	/** kde 'pkptr_prev' je paket ktery je momentalne prijiman **/
		FIN_MT (dra_inoise (pkptr_prev, pkptr_arriv));

  /* Jestlize kdyz dorazi novy paket predchozi paket prijem dokoncil, */
  /* nejedna se o kolizi. (tesne minuti paketu nebo back-to-back pakety) */
	if (op_td_get_dbl (pkptr_prev, OPC_TDA_RA_END_RX) != op_sim_time ())
		{

    /* Prirustek poctu kolizi v predcozim paketu. */
		op_td_increment_int (pkptr_prev, OPC_TDA_RA_NUM_COLLS, 1);

    /* Prirustek poctu kolizi v prichozim paketu. */
		op_td_increment_int (pkptr_arriv, OPC_TDA_RA_NUM_COLLS, 1);

    /* Urceni, zda predchozi paket je platny neo sum */
		prev_match = op_td_get_int (pkptr_prev, OPC_TDA_RA_MATCH_STATUS);

    /* Urceni, zda prichozi paket je platny neo sum */
		arriv_match = op_td_get_int (pkptr_arriv, OPC_TDA_RA_MATCH_STATUS);

    /* Jestlize je prichazejici paket platny, vypocitej.   */
    /* inteferenci predchoziho paketu na prichazejici paket       */
		if (arriv_match == OPC_TDA_RA_MATCH_VALID)
			{
			prev_rcvd_power   = op_td_get_dbl (pkptr_prev,
OPC_TDA_RA_RCVD_POWER);
			op_td_increment_dbl (pkptr_arriv, OPC_TDA_RA_NOISE_ACCUM,
prev_rcvd_power);
			}

    /* Jestlize je predchazejici paket platny, vypocitej.  */
    /* inteferenci prichoziho paketu na predchazejici paket       */
		if (prev_match == OPC_TDA_RA_MATCH_VALID)
			{
			arriv_rcvd_power = op_td_get_dbl (pkptr_arriv,
OPC_TDA_RA_RCVD_POWER);
			op_td_increment_dbl (pkptr_prev, OPC_TDA_RA_NOISE_ACCUM,
arriv_rcvd_power);
			}
	FOUT
	}
```

PŘÍLOHA K Kód modelu 9.kroku dra_bkgnoise

```
/* dra_bkgnoise.ps.c */
/* Vychozi model sumu pozadi radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009   */
/*           by OPNET Technologies, Inc.*/
/*           (A Delaware Corporation)    */
/*     7255 Woodmont Av., Suite 250      */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.*/
/*****************************************/

#include "opnet.h"

/***** konstanty *****/
#define BOLTZMANN                1.379E-23
#define BKG_TEMP                 290.0
#define AMB_NOISE_LEVEL          1.0E-26

/***** procedury *****/
#if defined (__cplusplus)
extern "C"
#endif
void
dra_bkgnoise_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       double       rx_noisefig, rx_temp, rx_bw;
       double       bkg_temp, bkg_noise, amb_noise;

  /** Vypocet sumu pozadi. **/
       FIN_MT (dra_bkgnoise (pkptr));

  /* Nacteni sumoveho cinitele prijimace. */
       rx_noisefig = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_NOISEFIG);

  /* Vypocet efektivni teplot prijimace . */
       rx_temp = (rx_noisefig - 1.0) * 290.0;

  /* Nacteni efektivni teploty pozadi. */
       bkg_temp = BKG_TEMP;

  /* Nacteni sirky pasma prijmaciho kanalu (v Hz). */
       rx_bw = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_BW);

  /* Vypocet pasmoveho sumu pozadi z sumu pozadi a tepelneho sumu . */
       bkg_noise = (rx_temp + bkg_temp) * rx_bw * BOLTZMANN;

  /* Vypocet pasmoveho sumu okoli z urovne sumu okoli . */
       amb_noise = rx_bw * AMB_NOISE_LEVEL;

  /* Nastaveni prislusneho TDA na hodnotu souctu pasmovych sumu pozadi a okoli .*/
       op_td_set_dbl (pkptr, OPC_TDA_RA_BKGNOISE, (amb_noise + bkg_noise));

       FOUT
       }
```

PŘÍLOHA L Kód modelu 10.kroku dra_snr

```
/* dra_snr.ps.c */
/* Vychozi model kroku Signal-to-Noise-Ratio (SNR) radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009 * /
/*           by OPNET Technologies, Inc.*/
/*           (A Delaware Corporation)   */
/*     7255 Woodmont Av., Suite 250     */
/*      Bethesda, MD 20814, U.S.A.      */
/*                  All Rights Reserved.*/
/*****************************************/

#include "opnet.h"
#include <math.h>

#if defined (__cplusplus)
extern "C"
#endif
void
dra_snr_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       double       bkg_noise, accum_noise, rcvd_power;

  /** Vypocet SNR pro dany paket. **/
       FIN_MT (dra_snr (pkptr));

  /* Nacti vykon prijateho signalu paketu. */
       rcvd_power = op_td_get_dbl (pkptr, OPC_TDA_RA_RCVD_POWER);

  /* Nacti sumove urovne paketu pocitane v krocich interferencniho sumu a  */
  /* sumu pozadi. */
       accum_noise = op_td_get_dbl (pkptr, OPC_TDA_RA_NOISE_ACCUM);
       bkg_noise = op_td_get_dbl (pkptr, OPC_TDA_RA_BKGNOISE);

  /* Uloz SNR v dB. */
       op_td_set_dbl (pkptr, OPC_TDA_RA_SNR,
              10.0 * log10 (rcvd_power / (accum_noise + bkg_noise)));

  /* V prislusnem TDA nastav cas ve kterem byl SNR pcitan. */
       op_td_set_dbl (pkptr, OPC_TDA_RA_SNR_CALC_TIME, op_sim_time ());

       FOUT
       }
```

PŘÍLOHA M Kód modelu 11.kroku dra_ber

```
/* dra_ber.ps.c */
/* Vychozi model Bit-Error-Rate (BER) radioveho prenosu */

/*****************************************/
/*             Copyright (c) 1993-2009   */
/*           by OPNET Technologies, Inc.*/
/*           (A Delaware Corporation)    */
/*     7255 Woodmont Av., Suite 250      */
/*      Bethesda, MD 20814, U.S.A.       */
/*                  All Rights Reserved.*/
/*****************************************/

#include "opnet.h"

#if defined (__cplusplus)
extern "C"
#endif

void
dra_ber_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
       {
       double       ber, snr, proc_gain, eff_snr;
       Vartype             modulation_table;

/* Vypocet prumerne bitove chybovosti ovlivnujici dany paket. */
       FIN_MT (dra_ber (pkptr));

/* Nacteni aktualni hodnoty Signal-to-Noise-Ratio (SNR). */
       snr = op_td_get_dbl (pkptr, OPC_TDA_RA_SNR);

/* Nacteni ukazatele na modulacni tabulku. */
       modulation_table = op_td_get_ptr (pkptr, OPC_TDA_RA_RX_MOD);

/* Nacteni zisku prijimaciho systemu. */
       proc_gain = op_td_get_dbl (pkptr, OPC_TDA_RA_PROC_GAIN);

/* Vypocet efektivniho SNR zahrnujici zpracovani zisku. */
       eff_snr = snr + proc_gain;

/* Odvozeni ocekavaneho BER z efektivniho SNR. */
       ber = op_tbl_mod_ber (modulation_table, eff_snr);

/* Nastaveni prislusneho TDA na hodnotu BER. */
       op_td_set_dbl (pkptr, OPC_TDA_RA_BER, ber);

       FOUT
       }
```

PŘÍLOHA N Kód modelu 12. kroku dra_error

```
/* dra_error.ps.c */
/* Vychozi model alokace chyb radioveho prenosu. Model ukoncuje vypocet */
/* konecneho poctu bitovych chyb paketu ve chvili, kyz je zjisteno, ze  */
/* pocet bitovych chyb jiz presahuje prah opravitelnosti prijimace, coz */
/* znamena, ze prijimac tak jako tak paket odmitne. Takze i tento model */
/* vice chyb nehleda. proto tento model nabizi rychlejsi zpracovani koku */
/* bez vlivu na chovani modelu. Statistiky "number of bit errors" */
/* a "actual bit-error-rate" pro kazdy paket nemohou byt presne   */
/* Pro simulace, ve kterych pozadujeme tyto informace je treba pouzit   */
/* "*_all_stats" model tohoto kroku    */


/*****************************************/
/*            Copyright (c) 1993-2009    */
/*           by OPNET Technologies, Inc.*/
/*            (A Delaware Corporation)   */
/*        7255 Woodmont Av., Suite 250   */
/*      Bethesda, MD 20814, U.S.A.        */
/*              All Rights Reserved.     */
/*****************************************/

#include "opnet.h"
#include <math.h>

/* Definice usnadnujiciho makra pro vypocet faktorialu uzitim gamma funkce     */
/* pro mt-safe verze pro paralelni spousteni, ktere je pristupne pouze    */
/* u systemu Solaris.*/

#if defined (OPD_PARALLEL) && !defined (HOST_PC_INTEL_WIN32)
#define log_factorial(n)         lgamma_r ((double) n + 1.0, &signgam)
extern double lgamma_r (double, int *);
#else
#define log_factorial(n)         lgamma ((double) n + 1.0)
extern double lgamma (double);
#endif

#define round(x) (floor (x + 0.5))

#if defined (__cplusplus)
extern "C"
#endif
void
dra_error_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet* pkptr)
      {
      double              pe, r, p_accum, p_exact;
      double              data_rate, elap_time;
      double              log_p1, log_p2, log_arrange;
      double              ecc_thresh;
      double              pklen;
      OpT_Packet_Size     seg_size;
      int                       num_errs, prev_num_errs;
#if defined (OPD_PARALLEL) && !defined (HOST_PC_INTEL_WIN32)
      int                       signgam;
#endif

  /** Vypocet poctu bitovych chyb oznacene pro segment bitu v paketu **/
  /** zalozey na jeho delce a pravdepodobnosti chyb  **/
      FIN_MT (dra_error (pkptr));

  /* Nacteni celkoveho poctu bitovych chyb, ktere jsou jiz oznaceny   */
  /* v predchozich segmentech paketu */
      prev_num_errs = op_td_get_int (pkptr, OPC_TDA_RA_NUM_ERRORS);

  /* Jestlize pocet bitovych chyb v paketu jiz presahuje prahovou hodnotu ecc*/
  /* tak neni treba dale overovat, zda existuji dalsi chyby, protoze paket     */
  /* bude jz tak jako tak odmitnut. Nacteni velikosti paketu a prah     */
```

```
/* opravitelnosti je provedeno pouze pokud paket jiz chyby obsahuje   */
    if (prev_num_errs > 0)
          {
          ecc_thresh = op_td_get_dbl (pkptr, OPC_TDA_RA_ECC_THRESH);
          pklen      = (double) op_pk_total_size_get (pkptr);

  /* Kontrola, zda jiz pocet chyb prekrocil prah  */
          if ((double) prev_num_errs / pklen > ecc_thresh)
                {
                FOUT;
                }
          }
    else
          {

  /* Nastaveni delky paketu na neplatnou hodnotu pro indikaci,        */
  /* ze nebyl prijat    */
          pklen = -1.0;
          }

/* Nacteni ocekavaneho BER 'pe'.     */
    pe = op_td_get_dbl (pkptr, OPC_TDA_RA_BER);

/* Vypcet doby, ktera uplynula od posledni zmeny BER     */
    elap_time = op_sim_time () - op_td_get_dbl (pkptr, OPC_TDA_RA_SNR_CALC_TIME);

                */
/* Pouzijeme rychosl prenosu dat ke zjisteni poctu bitu v segmentu    */
    data_rate = op_td_get_dbl (pkptr, OPC_TDA_RA_RX_DRATE);
    seg_size = (OpT_uInt64) round (elap_time * data_rate);


/* Pripad 1: Jestlize BER je 0, tak je i nula bitovych chyb     */
  if (pe == 0.0 || seg_size == 0)
          num_errs = 0;

/*   Pripad 2: Jestlize BER se rovna 1.0, tak vsechny bity jsou chybne. */
/* Zpravidla vsak BER neprekroci hodnotu 0.5*/

    else if (pe >= 1.0)
          num_errs = seg_size;

/* Pripad 3: BER neni ani 0 ani 1 */
    else
          {

  /* Pocet chyb lze ziskat tak, ze bude definovano nahodne cislo v    */
  /* intervalu (0,1) pomoci inverzni funkce k distribucni fce psti (CMF)     */
  /* pro rozlozeni poctu bitovych chyb      */

  /* Ziskani nahodneho cisla v intervalu (0,1) jako vysledek CMF      */
          r = op_dist_uniform (1.0);

  /* Iterace rozlozeni pravdepodobnosti nad moznymi vysledky az do    */
  /* prekroceni hodnoty r. Smycka opakovane odpovida obracenim CMF    */
  /* jakmile najde pocet bitovych chyb, pri kterem CMF poprve dosahne */
  /* nebo prekraci hodnotu r.         */
          for (p_accum = 0.0, num_errs = 0; num_errs <= seg_size; num_errs++)
                {

        /* Vypocet pravdepodobnosti presneho vyskytu bitovych chyb 'num_errs'      */

    /* Pravepodobnost, ze prvni 'num_errs' bitu bude chybne je dana */
    /* pow(pe,num_errs). Zde je ziskana v logaritmicke podobe, aby */
    /* nedoslo k podteceni pro male 'pe' nebo velke 'num_errs'. */
                log_p1 = (double) num_errs * log (pe);
    /* Stejne tak lze ziskat pravdepodobnost, ze zbyvajici bity       */
    /* nebudou chybne. Kombinace techto dvou pripadu vytvari jednu moznou    */
```

```
    /* konfiguraci bitu prinasejici celkovy pocet bitovych chyb 'num_errs'   */
                log_p2 = (double) (seg_size - num_errs) * log (1.0 - pe);

    /* Vypocet mozneho usporadani se stejnym poctem bitovych chyb jako  */
    /* v pripade vyse. Opet je toto cislo vyjadreno logaritmicky   */
    /* z duvodu preteceni. Tento vysledek je vyjadren v logaritmicke  */
    /* podobe ze vztahu pro hodnotu N z kombinaci k z n: N = n!/(n-k)!k! */
                log_arrange =       log_factorial (seg_size) -
                                          log_factorial (num_errs) -
                                          log_factorial (seg_size - num_errs);

    /* Vypocet pravdepodobnosti, ze presne 'num_errs' je pritomno v   */
    /* segmentu ve vsech usporadanich       */
                p_exact = exp (log_arrange + log_p1 + log_p2);

    /* Pridani do rozlozeni pravdepodobnosti akumulovane pro zatim drive */
    /* testovane vysledky k ziskani hodnoty vysledku CMF jako num_errs .*/
                p_accum += p_exact;

    /* 'num_errs' je vysledkem tohoto procesu v pripade, ze CMF splnuje*/
    /* nebo prekracuje jednotnou nahodnou hodnotu urcenou drive */
                if (p_accum >= r)
                      break;

    /* Bude-li dosazeno tohoto bodu, pak paket ma alespon jednu bitovou      */
    /* chybu, ktera jiz muze postacovat k prekroceni prahove hodnoty ECC.    */
    /* Pokud tomu tak je, vypocet bude zastaven, protoze vime, ze paket      */
    /* bude tak jako tak odmitnut. Pokud se jedna o uplne prvni bitove       */
    /* chyby, potom nacteme celkovou velikost paketu a prahovou hodnotu      */
    /* ktere potrebujeme pro porovnani nize */
                if (pklen < 0.0)
                      {
                      ecc_thresh = op_td_get_dbl (pkptr, OPC_TDA_RA_ECC_THRESH);
                      pklen      = (double) op_pk_total_size_get (pkptr);
                      }

    /* Kontrola, zda celkovy pocet chyb jiz prekrocil prah.     */
                if ((double) (prev_num_errs + num_errs + 1) / pklen > ecc_thresh)
                      {

      /* Prirustek poctu bitovych chyb, ktery bychom provedli v pripade, */
      /* ze budeme pokracovat v hledani vysiho poctu bitovych bych*/
                      num_errs++;

                      /* Terminate the for-loop.       */
      /* Konec cyklu*/
                      break;
                      }
                }

/* Navyseni poctu bitovych chyb v prislusnem TDA */
    op_td_set_int (pkptr, OPC_TDA_RA_NUM_ERRORS, num_errs + prev_num_errs);

/* Prirazeni aktualni BER na testovanem segmentu paketu */
    if (seg_size != 0)
          op_td_set_dbl (pkptr, OPC_TDA_RA_ACTUAL_BER, (double) num_errs / seg_size);
    else op_td_set_dbl (pkptr, OPC_TDA_RA_ACTUAL_BER, pe);

    FOUT
    }
```

PŘÍLOHA O Kód modelu 13. kroku dra_ecc

```
/* dra_ecc.ps.c */
/* Vychozi model korekce chyb radioveho prenosu. */
/* Tento model vyuziva stavove informace prijimaciho kanalu*/
/* pro aktualizaci zablokovani prijmu. To zalezi na kroku modelu */
/* rxgroup, kdy vytvori a inicializuje stavove informace kanalu */

/*****************************************/
/*            Copyright (c) 1993-2009  */
/*           by OPNET Technologies, Inc.*/
/*            (A Delaware Corporation)  */
/*        7255 Woodmont Av., Suite 250   */
/*      Bethesda, MD 20814, U.S.A.       */
/*              All Rights Reserved.     */
/*****************************************/

#include "opnet.h"
#include "dra.h"

#if defined (__cplusplus)
extern "C"
#endif

void
dra_ecc_mt (OP_SIM_CONTEXT_ARG_OPT_COMMA Packet * pkptr)
    {
      int                              num_errs, accept;
      OpT_Packet_Size                  pklen;
      double                           ecc_thresh;
      DraT_Rxch_State_Info*     rxch_state_ptr;

  /** Urceni prijatelnosti paketu na prijmu. **/
      FIN_MT (dra_ecc (pkptr));

  /* Neprijimat pakety neaktivniho uzlu */
      if (op_td_is_set (pkptr, OPC_TDA_RA_ND_FAIL))
            accept = OPC_FALSE;
      else
            {

    /* Nacteni prahove hodoty poctu chyb prijimace. */
            ecc_thresh = op_td_get_dbl (pkptr, OPC_TDA_RA_ECC_THRESH);

    /* Nacteni delky paketu. */
            pklen = op_pk_total_size_get (pkptr);

    /* Nacteni poctu chyb v paketu . */
            num_errs = op_td_get_int (pkptr, OPC_TDA_RA_NUM_ERRORS);

    /* Test bitovych chyb na prahovou hodnotu . */
            if (pklen == 0)
                  accept = OPC_TRUE;
            else
                  accept = ((((double) num_errs) / pklen) <= ecc_thresh) ? OPC_TRUE:
OPC_FALSE;
            }

  /* Nastaveni prislusneho TDA indikujiciho prijatelnost paketu. */
      op_td_set_int (pkptr, OPC_TDA_RA_PK_ACCEPT, accept);

  /* Pro oba pripady jiz kanal nebude blokovan. */
      rxch_state_ptr = (DraT_Rxch_State_Info *) op_ima_obj_state_get (op_td_get_int
(pkptr, OPC_TDA_RA_RX_CH_OBJID));
      rxch_state_ptr->signal_lock = OPC_FALSE;

      FOUT
      }
```