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

DYNAMICKÝ SMĚROVACÍ PROTOKOL OSPFV3 PRO MANET SÍTĚ V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

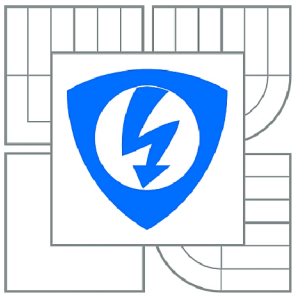
AUTOR PRÁCE
AUTHOR

JÚLIUS HENZELY

BRNO 2011

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

DYNAMICKÝ SMĚROVACÍ PROTOKOL OSPFV3 PRO MANET SÍTĚ V SIMULAČNÍM PROSTŘEDÍ OPNET MODELER

OSPFV3 DYNAMIC ROUTING PROTOCOL FOR MANET NETWORKS IN OPNET MODELER SIMULATION ENVIRONMENT

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

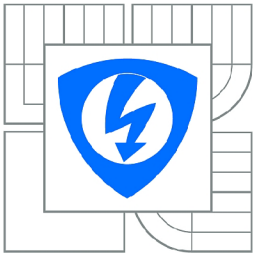
AUTOR PRÁCE — JÚLIUS HENZELY

AUTHOR

VEDOUCÍ PRÁCE — Ing. LUKÁŠ RŮČKA

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Július Henzely	***ID:***	106461
Ročník:	3	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Dynamický směrovací protokol OSPFv3 pro MANET sítě v simulačním prostředí OPNET Modeler

POKYNY PRO VYPRACOVÁNÍ:

Seznamte se s problematikou MANET sítí. Zaměřte se zejména procesy směrování v těchto sítích. Seznamte se s protokolem IPv6. Proveďte podrobný teoretický rozbor směrovacího protokolu OSPFv3 a jeho srovnání s protokolem OSPFv2. Seznamte se se simulačním prostředím OPNET Modeler a jeho možnostmi konfigurace a simulace modelu MANET sítě. Prostudujte a zdokumentujete procesní model směrovacího protokolu OSPFv3 v prostředí OPNET Modeler. Identifikujte místa a možnosti úpravy obsahu směrovacích zpráv tohoto protokolu. Upravte či rozšiřte původní zprávy směrovacího protokolu o další pole, pomocí kterého může bezdrátová stanice informovat své okolí o míře využití dostupné přenosové kapacity. Upravte původní mechanismy zpracování zpráv směrovacího protokolu OSPFv3 o vyhodnocení a ukládání zatížení sousedních uzlů, pravidelnou aktualizaci těchto údajů a aktualizaci údajů o stanicích, které jsou v dosahu. Všechna svá zjištění a provedené úpravy podrobně zdokumentujte a přehledně prezentujte.

DOPORUČENÁ LITERATURA:

[1] Doyle, J., Carroll, J.: Routing TCP/IP. Indianopolis: Cisco Press, 2005. 936 s. ISBN: 1-58705-202-4.
[2] OPNET Technologies, OPNET Modeler Product Documentation Release 16.0, OPNET Technologies Inc., 2010.

Termín zadání:	7.2.2011	***Termín odevzdání:***	2.6.2011

Vedoucí práce: Ing. Lukáš Růčka

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

ABSTRAKT

V bakalárskej práci sú zahrnuté teoretické poznatky týkajúce sa základných znalostí ohľadom funkcie mobilných sietí so zameraním na siete MANET a smerovacie protokoly v nich použité. Práca sa zaoberá prehľadom smerovacích protokolov, predovšetkým smerovacím protokolom OSPFv3. V praktickej časti práce je stručný popis simulačného prostredia OPNET Modeleru, postup zostavenia siete a konfigurácia smerovacieho protokolu v nej. Súčasťou práce je aj postup rozšírenia správy Hello o nové pole na prenos rýchlosti uzla v OPNET Modeleri.

KLÍČOVÁ SLOVA

Ad hoc siete, MANET, mobilné siete, OPNET Modeler, OSPFv3, smerovacie protokoly, smerovanie, Hello paket

ABSTRACT

This bachelor's thesis includes theoretical knowledge about fundamental informations related to functionality of mobile networks, with focus on MANET routing protocols. The project contains an overview of routing protocols, especially OSPFv3 routing protocol. In the practical section of project, there is a brief description of OPNET Modeler simulation environment and the process of establishing the network with configured routing protocol in it. The project also describes a procedure for extending of the Hello packet structure by a field containing node traffic speed in OPNET Modeler.

KEYWORDS

Ad hoc networks, MANET, mobile networks, OPNET Modeler, OSPFv3, routing protocols, routing, Hello packet

HENZELY, Július *Dynamický směrovací protokol OSPFv3 pro MANET sítě v simulačním prostředí OPNET Modeler*: bakalářská práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 69 s. Vedoucí práce byl Ing. Lukáš Růčka,

PROHLÁŠENÍ

Prohlašuji, že svou bakalářskou práci na téma „Dynamický směrovací protokol OSPFv3 pro MANET sítě v simulačním prostředí OPNET Modeler“ jsem vypracoval samostatně pod vedením vedoucího bakalářské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené bakalářské práce dále prohlašuji, že v souvislosti s vytvořením této bakalářské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

Brno ....

(podpis autora)

Na tomto mieste by som sa chcel poďakovať najmä vedúcemu môjej bakalárskej práce pánovi Ing. Lukášovi Růčkovi, za jeho pripomienky a rady ohľadom riešenia tejto bakalárskej práce. Ďalej by som chcel poďakovať pánovi docentovi Karolovi Molnárovi, Ing. Jiřímu Hoškovi a Ing. Pavlovi Vajsarovi za ich čas počas konzultácií. Tiež by som chcel poďakovať firme OPNET Technologies Inc. za technickú podporu ku programu OPNET Modeler.

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

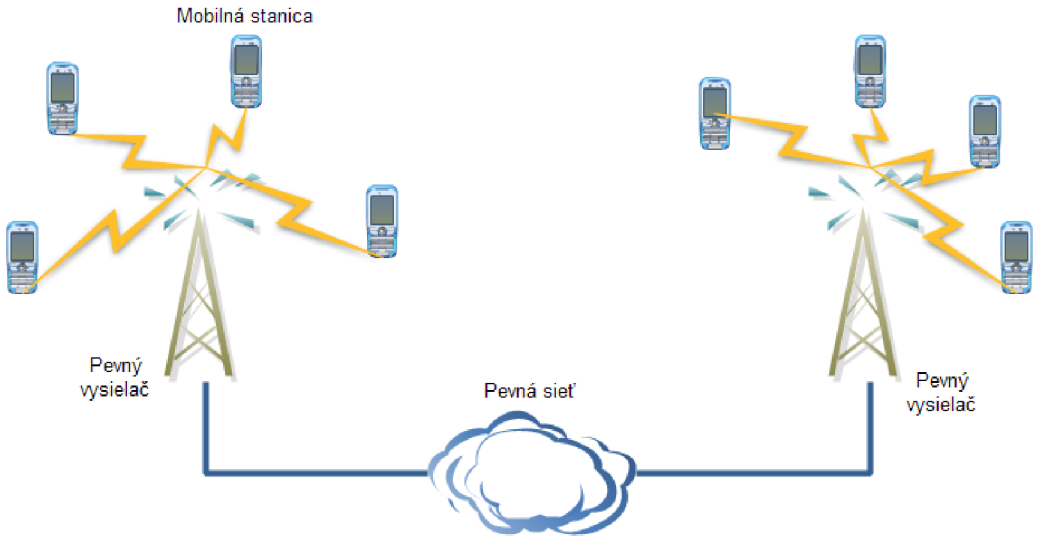
V súčasnosti sú klasické bezdrôtové siete bežnou záležitosťou vďaka obrovskému rozšíreniu rôznych mobilných zariadení, najmä notebookov a netbookov. Preto vznikla myšlienka vytvoriť sieť pre takéto mobilné zariadenia. Klasické bezdrôtové siete väčšinou dovoľujú len obmedzenú pohyblivosť staníc v okolí prístupového bodu. Riešenie poskytujú mobilné Ad hoc siete (Mobile Ad hoc Networks MANET). MANET siete poskytujú oproti klasickej bezdrôtovej sieti jednu veľkú výhodu, ktorou je nezávislosť od akejkoľvek infraštruktúry [11].

Pre siete MANET je vyvíjaných množstvo sieťových protokolov. Väčšiu časť z nich vyvíja skupina MANET Working Group organizácie IETF. Smerovací protokol OSPFv3 je však vyvíjaný firmou Cisco Systems, Inc. [11]. Rozdiel oproti ostatným smerovacím protokolom pre MANET siete je ten, že OSPFv3 je postavený na základoch, ktoré sa používali aj u káblových sietí. Ďalšou spoločnosťou, ktorá podporuje protokol OSPFv3, je Boeing.

Hlavným cieľom tejto bakalárskej práce je oboznámenie sa so sieťami MANET, smerovacím protokolom OSPFv3 a simulačným prostredím OPNET Modeler (OM).

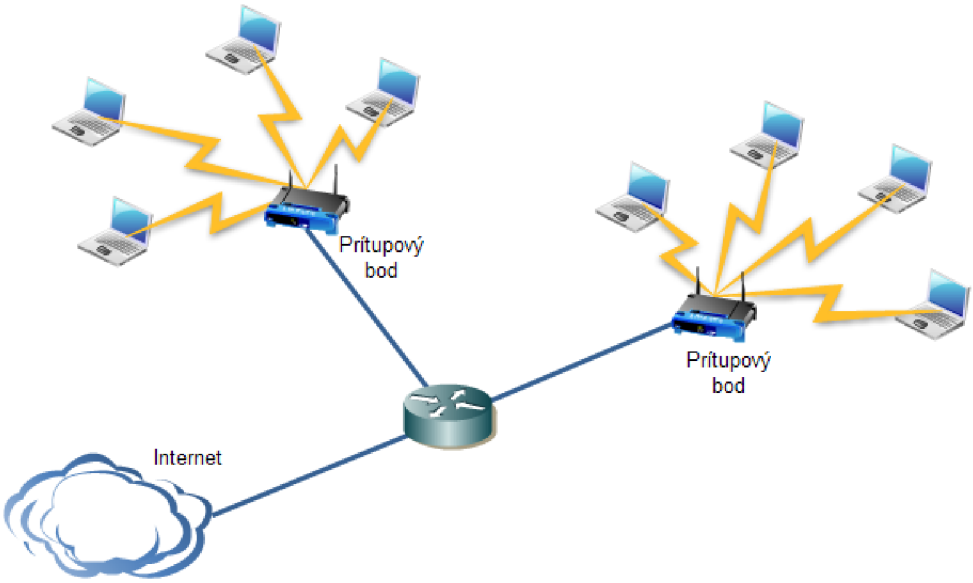
1 MOBILNÉ SIETE

V súčasnosti sa dajú rozdeliť mobilné siete na tri základné druhy. Prvým typom mobilných sietí sú tradičné siete pre mobilnú komunikáciu napr. GSM (Global System for Mobile Communications). Základom tejto siete sú pevné vysielače, v ktorých dosahu sú mobilné stanice dostupné [8]. Dosah vysielača je niekoľko kilometrov a riadenie siete sa odohráva na jednom centrálnom mieste. Nevýhodami tradičnej mobilnej siete sú najmä nutnosť zriadenia infraštruktúry a veľký výstupný výkon vysielača a taktiež staníc.



Obr. 1.1: Tradičná mobilná sieť

Druhým typom sú bežné bezdrôtové siete. Jedná sa zvyčajne o sieť, ktorej centrálnym prvkom je bezdrôtový prístupový bod, ku ktorému sa pripájajú jednotlivé mobilné stanice. Samotný prístupový bod je väčšinou pripojený k pevnej sieti pomocou kábla [8]. Hlavnou výhodou bezdrôtových sietí je ich nízka zriaďovacia cena a redukcia nutnej kabeláže v danej oblasti. Nevýhodami sú najmä malé územie pokryté signálom a nutnosť zabezpečenia aspoň základnej infraštruktúry v oblasti. Avšak potrebná infraštruktúra je znateľne jednoduchšia ako v prípade klasických sietí pre mobilné telefóny.



Obr. 1.2: Sieť s bezdrôtovým prístupom

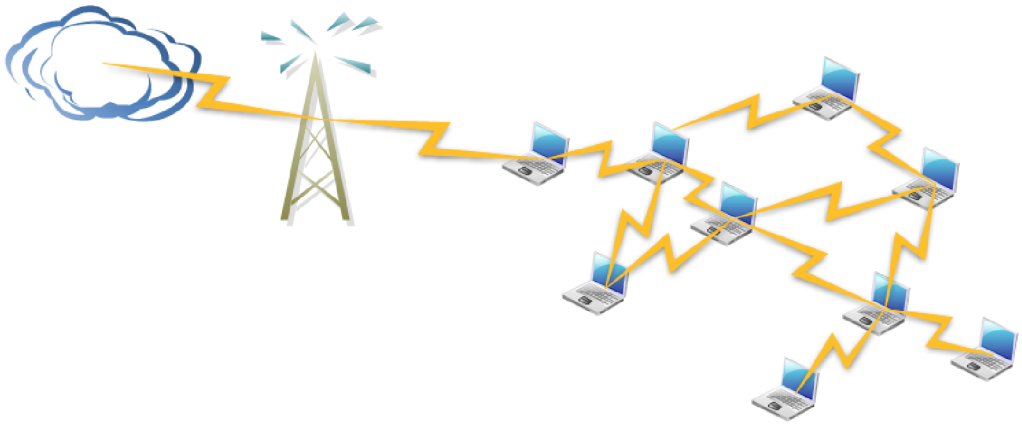
Posledným druhom sú mobilné Ad hoc siete. Týmito sieťami sa bude detailnejšie zaoberať práve nasledujúca kapitola.

1.1 MANET

MANET je sieť, v ktorej neexistuje žiadna centrálna infraštruktúra. Sieť tvoria mobilné uzly, ktoré samé určujú a spravujú smerovanie, zatiaľ čo sa samé pohybujú a v pohybe sú aj ich komunikačné ekvivalenty. Podobné siete boli v minulosti nazývané ako siete typu Packet Radio a sú v obmedzenej miere využívané vo vojenskom nasadení. [11] S obrovským rozvojom mobilných bezdátových technológií získavajú siete MANET stále väčšie vojenské i komerčné uplatnenie.

Sieť MANET je autonómna skupina uzlov rozptýlená po určitej geografickej oblasti, ktoré komunikujú prostredníctvom bezdrôtových spojov s obmedzenou šírkou pásma. V podstate sa jedná len o skupinu uzlov, ktoré chcú komunikovať bez existujúcej infraštruktúry, pričom sa môže jednať o vojenskú jednotku, sieť mobilných senzorov či účastníkov konferencie. Každý uzol v sieti MANET v určitom momente predstavuje vysielač, prijímač alebo predávajúcu stanicu s rôznymi fyzickými a vysielacími schopnosťami. Podstatnou vlastnosťou je napájací zdroj, pretože veľa uzlov môže pracovať len z obmedzeného napájania z batérie. V sieťach MANET pakety väčšinou prechádzajú niekoľkými predávajúcimi uzlami, ktoré ležia medzi vysielacou a cieľovou stanicou. Uzly sú mobilné a neexistuje žiaden centrálny smerovač, takže

v sieti nie je žiadna pevná topológia. Napriek tomu, že je na uzly kladená väčšia zodpovednosť, táto technológia umožňuje zriadenie mobilnej siete bez predchádzajúceho plánovania a nutnosti zaobstarania infraštruktúry.



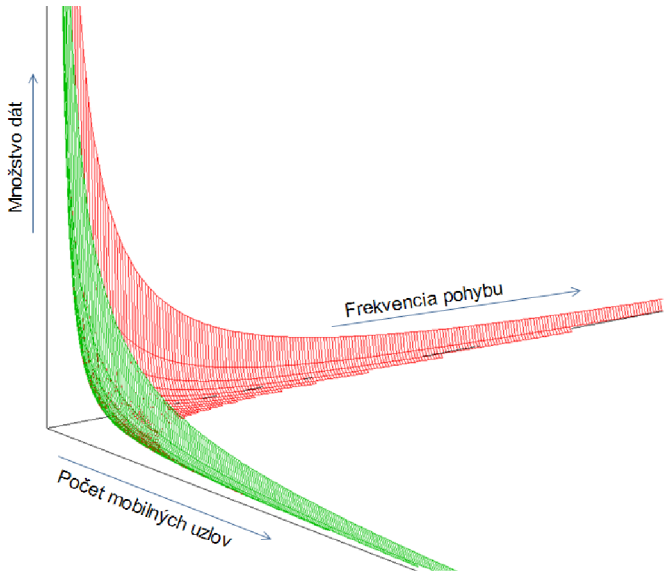
Obr. 1.3: Príklad siete MANET

Siete MANET trpia obvyklými problémami bezdrôtovej komunikácie, medzi ktoré patrí strata trasy šírenia, prenos signálu po niekoľkých trasách, kolísanie intenzity signálu a tepelný šum. Tieto podmienky sa menia s pohybom terminálov, ktorý tiež spôsobuje dopplerovský frekvenčný rozptyl vysielaného signálu. Ďalšie rušenie môže spôsobovať vysielanie susedných terminálov (interferencia pri viacnásobnom prístupe), ako tiež rušičky signálu a impulzné interferencie z rôznych zdrojov v rovnakom frekvenčnom pásme [11]. Trasy smerovania medzi dvoma uzlami siete sa líšia v závislosti na prostredí a výslednej topológii siete kvôli premenlivej kvalite a vlastnostiam spoja. Topológia siete môže byť dlhú dobu stabilná, ale dôsledkom mobility staníc sa môže nepredvídateľne meniť. Vzhľadom k absencii akýchkoľvek ústredných radičov a smerovačov všetky rozhodnutia o smerovaní predávaní paketov vykonávajú samotné uzly a komunikácia prebieha spôsobom peer-to-peer.

Mobilné Ad hoc siete majú množstvo variánt v závislosti na veľkosti a hustote siete, mobilite uzlov a komunikačnom prostredí. MANET s vysokou pohyblivosťou uzlov môže mať iné požiadavky na smerovanie ako MANET s obmedzenou frekvenciou zmien topológie, preto neexistuje nič také ako optimálny smerovací protokol [11].

Z chovania siete MANET sa dá vyvodiť niekoľko záverov. Na obrázku 4 na nachádza trojrozmerný graf, ktorý charakterizuje vlastnosti smerovania siete MANET. Tento graf prakticky ukazuje súvislosť medzi objemom prenášaných dát, počtom uzlov v sieti a frekvenciou pohybu uzlov. Z grafu vyplýva, že v sieti s veľkým počtom

pohyblivých uzlov môže byť prenesený menší objem dát ako v sieti s malým počtom uzlov, ktoré zostávajú v pokoji.



Obr. 1.4: Vlastnosti siete MANET [11]

Nie je možné, aby jeden protokol optimálne pracoval v každej situácii. Skupina MANET Working Group organizácie IETF sa problémom protokolov pre MANET zaoberá už niekoľko rokov, čoho výsledkom je množstvo navrhnutých smerovacích protokolov pre MANET siete. Tieto protokoly sa dajú rozdeliť do dvoch väčších skupín: Proaktívne protokoly a reaktívne protokoly (typu on-demand). Proaktívne protokoly sú charakteristické čakacou dobou na začiatku pri určovaní všetkých trás, hoci dané trasy sa nemusia počas fungovania siete ani raz použiť. Reaktívne protokoly naopak vykazujú dlhšie čakacie doby pri zisťovaní trasy, keď je trasa potrebná a tiež ak používajú trasu, ktorá je stará. Do fázy dokumentu RFC boli zatiaľ zaradené 4 z týchto protokolov:

- Ad hoc On Demand Distance Vector Routing (reaktívny)
- Dynamic Source Routing Protocol (reaktívny)
- Opimized Link State Routing (proaktívny)
- Rozosielanie topológie na základe predávania Reverse Path Forwarding (proaktívny)

Výkon každého z týchto protokolov závisí na konkrétnom prostredí, v ktorom je použitý. K reálnemu použitiu týchto protokolov zatiaľ nedošlo, preto bol ich výkon vyhodnotený len na základe testov. Podľa výsledkov sa dá urobiť záver, že protokoly typu on-demand pracujú lepšie v relatívne stabilných sieťach, kde sú stanovené

komunikačné dvojice zdroj – cieľ. Proaktívne protokoly naproti tomu vykazujú lepšie výsledky v prostredí s väčšou pohyblivosťou staníc, pretože je nevyhnutná réžia protokolu pri udržiavaní aktívnych smerovacích trás [11]. Na vlastnom protokole pre MANET siete pracuje aj spoločnosť Cisco, ktorá sa pokúša využiť už existujúce protokoly IOS, konkrétne OSPF (Open Shortest Path First)[2]. Použitie tohto postupu má výhodu najmä v tom, že sieť MANET sa dá následne jednoduchšie integrovať so sieťami s infraštruktúrou. Štandardizáciou smerovacieho protokolu OSPF-MANET bol v organizácii IETF poverený špeciálny tím, ktorého kľúčovým členom je práve spoločnosť Cisco [11].

Sieť MANET môže byť samostatná sieť, kde uzly komunikujú len medzi sebou (napríklad senzorová sieť), ale taktiež môže potrebovať komunikovať s globálnou sieťou Internet. Vtedy je nevyhnutné integrovať protokoly sietí MANET a Mobile IP. Obe siete dnes môžu existovať vedľa seba, pričom protokol Mobile IP (Internet Protocol) umožňuje globálnu komunikáciu prostredníctvom registrácie uzlu MANET do domovského agenta a protokol MANET komunikáciu typu peer-to-peer medzi uzlami.

2 SMEROVANIE

Smerovanie je proces, ktorý v sieťach zaisťuje výber dátovej cesty a jej následné použitie na prenos informácie. Najčastejšie je smerovanie spojené s prenosom v paketových sieťach preto, že vzhľadom k nutnosti smerovania každého paketu zvlášť sa táto činnosť vykonáva veľmi často. Dátové cesty sa získavajú zo smerovacej tabuľky, ktorá sa aktualizuje prostredníctvom vzájomnej výmeny smerovacích informácií o sieti s ostatnými uzlami v sieti [6]. K tomuto účelu sa využíva smerovací protokol. Efektivita smerovacieho protokolu závisí najmä na nasledujúcich vlastnostiach:

- presnosť – odosielanie paketov do správneho smeru
- spoľahlivosť – správna funkcia smerovania i pri vysokom zaťažení
- jednoduchosť hardwarovej implementácie
- rýchla konvergencia – rýchle ustálenie po zmenách v sieti
- flexibilita – podpora rôznych metrík
- použitie alternatívnych ciest

2.1 Smerovacie protokoly

Pre správne smerovanie dátových jednotiek je nutné, aby boli smerovače oboznámené s topológiou siete. Tieto informácie smerovač získa výmenou informácii s ostatnými smerovačmi (dynamické smerovanie) alebo konfiguráciou vykonanou správcom siete (statické smerovanie) [6]. Smerovače na komunikáciu využívajú smerovací protokol. Statické smerovanie so staticky nastavenými cestami pre svoju funkciu nepotrebuje využitie smerovacích protokolov. Jeho najväčšia nevýhoda je však to, že vôbec nereflektuje zmeny v sieti. Dynamické smerovanie automaticky detekuje zmeny v topológii siete a informácie o zmenách rozosiela automaticky ostatným smerovačom, aby svoje tabuľky aktualizovali.

Prvým typom smerovacích protokolov sú protokoly typu vektor vzdialeností (distance vector). Cesta sa pri smerovaní s použitím vektoru vzdialeností vyberá podľa najmenšieho počtu preskokov na ceste od zdroja k cieľu. Druhý typ smerovacích protokolov sa označuje ako protokoly stavu linky (link state). U tohto typu protokolov sa každej linke pripojenej k smerovaču pridelí tzv. metrika. Metrika určuje vlastnosti danej linky, ktoré má smerovač uložené v smerovacej tabuľke. Informácie o metrike svojich liniek smerovač rozpošle svojim susedom prostredníctvom správ LSA – Link State Advertisement. Ako najlepšia cesta sa potom vyberá tá, ktorá má súčet metrík najmenší [6].

2.1.1 Smerovacie protokoly typu vektor vzdialeností

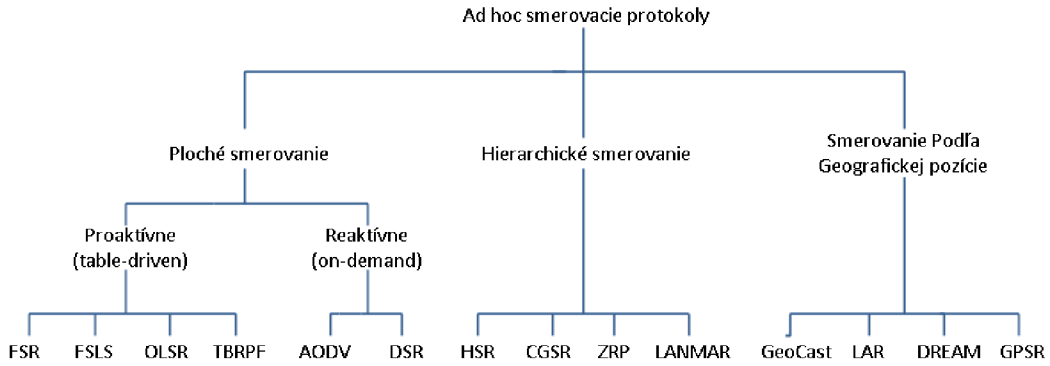
Základ týchto smerovacích protokolov spočíva vo výpočte najlepšej trasy. Tento výpočet vykonáva každý smerovač nezávisle na ostatných smerovačoch v sieti. Po skončení výpočtu smerovač rozpošle výsledky susedným smerovačom. Susedné smerovače potom zhodnotia svoje výsledky a v prípade nálezu lepšej cesty opäť túto informáciu rozpošlú všetkým susedom vrátane toho, od koho prišla pôvodne správa. Proces stanovenia najlepšej trasy prebieha vo viacerých cykloch a je dôležité, aby sa po určitom počte opakovaní zastavil a zabránil stavu, keď si smerovače neustále vymieňajú správy. Ideálne je, ak je počet týchto cyklov čo najmenší. Najlepšia cesta je v prípade týchto protokolov cesta s najmenším počtom preskokov [6]. Medzi výhody protokolov vektoru vzdialeností patrí jednoduchosť implementácie, údržby a riešenia prípadných problémov. Nevýhodou je obmedzený počet preskokov v sieti, čo zabraňuje použitiu protokolov vektoru vzdialeností vo väčších sieťach. Medzi protokoly vektoru vzdialeností patria napr. protokoly RIP a RIP2.

2.1.2 Smerovacie protokoly typu stav linky

Namiesto počítania preskokov sa pracuje s metrikami liniek. Ako najlepšia cesta je zvolená tá, ktorá má najmenší súčet metrík. Smerovač svojich susedov informuje o hodnotách metrík k nemu pripojených liniek pomocou Link State Advertisment (LSA). Správy o metrike sa posielajú všade okrem smeru, z ktorého boli prijaté. Smerovacie tabuľky sa aktualizujú len v prípade, ak je súčet metrík danej cesty v LSA menší ako aktuálna hodnota v tabuľke. O zmene vo svojej tabuľke informuje ostatné smerovače opäť pomocou LSA. V ustálenom stave majú smerovače rovnaké informácie a poznajú celú topológiu. Rozposielaním LSA na všetky porty, okrem toho odkiaľ bol prijatý, smerovač môže prijať LSA aj viackrát. Aby nedošlo k nekonečnému rozosielaniu LSA, musí byť smerovač schopný rozpoznať LSA, ktoré už videl. Problém sa rieši pomocou poradového čísla správy, hodnoty kontrolného súčtu alebo využitím poľa TTL (Time to Live). Optimálna cesta sa získa ako výsledok spracovania údajov algoritmom Shortest Path First (SPF). Metriky ciest môžu byť upravené vzhľadom ku podporovanej šírke pásma, podľa latencie alebo podľa úrovne zabezpečenia. Na označenie ciest sa najčastejšie využíva Dijkstrov algoritmus, ktorý hľadá cesty od zdrojového uzla ku všetkým ostatným uzlom v sieti [2].

2.2 Smerovacie protokoly použité v MANET sieťach

Rastúci záujem o siete typu Mobile Ad hoc viedol k návrhom mnohých smerovacích protokolov. Možnosti rozšírenia týchto sietí budia stále väčšiu pozornosť. V nasledujúcej kapitole je rozobrané rozdelenie a vlastnosti smerovacích protokolov sietí MANET. Smerovacie protokoly zahrnuté v prehľade spadajú do troch rôznych kategórií: plochých, hierarchických a geografických smerovacích protokolov.



Obr. 2.1: Rozdelenie smerovacích protokolov pre MANET [9]

2.2.1 Smerovanie v plochých sieťach

Protokoly spadajúce medzi ploché (flat) sa dajú rozdeliť do dvoch kategórií, menovite proaktívne a reaktívne (on-demand) [3]. Veľa proaktívnych protokolov pramení z konvenčných protokolov založených na stave linky (link state). Na druhej strane on-demand smerovanie je novo vznikajúca filozofia na poli Ad hoc sietí. Od konvenčných smerovacích protokolov sa líši najmä v tom, že pokiaľ v sieti nie je žiadna prevádzka, tak sa nevykonávajú žiadne smerovacie aktivity a ani žiadne smerovacie informácie sa neuchovávajú v sieťových uzloch a tým poskytuje riešenie pre husté siete. V nasledujúcej tabuľke sa nachádza zhrnutie vlastností protokolov FSR (Fisheye State Routing Protocol), OLSR (Optimized Link State Routing Protocol), TBRPF (Topology Broadcast based Reverse-Path Forwarding), AODV (Ad hoc On Demand Distance Vector Routing) a DSR (Dynamic Source Routing Protocol). Zložitosti jednotlivých protokolov sú udané ako Big O notácia a označujú funkcie z hľadiska náročnosti, napr. O (N) označuje lineárnu zložitosť a O (e) zložitosť exponenciálnu. Big O notácia označuje vlastnosti algoritmov z hľadiska výpočtovej náročnosti, priemerného alebo maximálneho času výpočtu alebo využitia pamäte.

Tab. 2.1: Porovnanie vlastností flat smerovacích protokolov [3]

	FSR	**OLSR**	**TBRPF**	**AODV**	**DSR**
Spôsob smerovania	proaktívny	proaktívny	proaktívny	reaktívny	reaktívny
Metrika smerovania	najkratšia cesta	najkratšia cesta	najkratšia cesta	najkratšia cesta	najkratšia cesta
Frekvencia aktualizácií	periodická	periodická	periodická, podľa potreby	podľa potreby	podľa potreby
Použitie sekvenčných čísel	áno	áno	áno (HELLO)	áno	áno
Zamedzenie slučkám	áno	áno	áno	áno	áno
Najhorší scenár	neexistuje	áno (uplný stav linky)	neexistuje	áno (úplné zaplavovanie)	áno(úplné zaplavovanie)
Použitie viacerých trás	áno	nie	nie	nie	áno
Pamäťová zložitosť	$O(N)$	$O(N)$	$O(N)$	$O(e)$	$O(e)$
Výpočtová zložitosť	$O(N)$	$O(N)$	$O(N)$	$O(N^2)$	$O(N^2)$

Proaktívne protokoly zdieľajú spoločný rys, ktorým je výmena smerovacích informácií na pozadí bez ohľadu na požiadavky siete. Obsahujú mnoho žiaducich vlastností najmä pre aplikácie na komunikáciu v reálnom čase, vrátane garancie QoS (Quality of Service), malého oneskorenia pri prístupe k smerovacej ceste a podpory a kontroly alternatívnej QoS cesty [3].

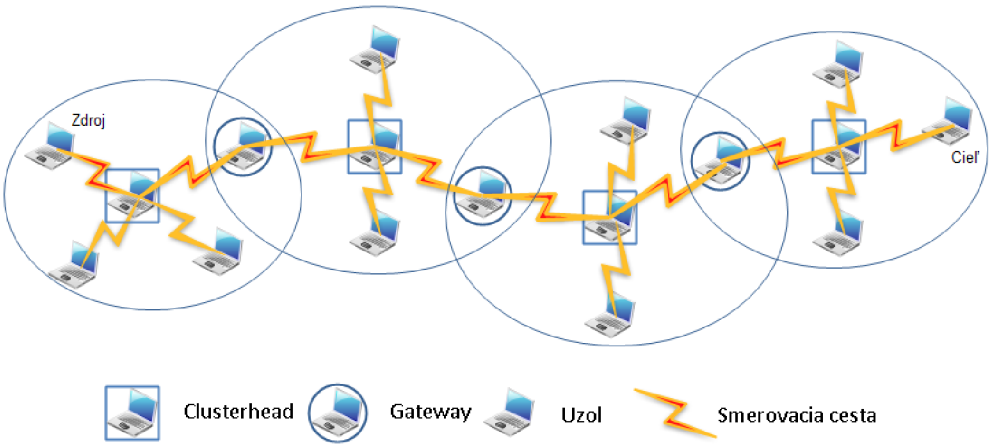
Reaktívne algoritmy väčšinou majú tzv. fázu objavovania cesty. Sieť sa zaplavuje query paketmi vyslanými zdrojmi pri hľadaní cesty. Fáza sa ukončí, keď je cesta k cieľu nájdená alebo sú vyhľadané všetky možné cesty v sieti. K hľadaniu ciest je v reaktívnych algoritmoch viacero prístupov. V AODV protokole sa po získaní query paketu tranzitné uzly učia cestu k zdroju (tzv. spätné učenie) a zadávajú ju do smerovacej tabuľky. Časom sa query paket dostane k cieľovej stanici a tá môže odpovedať prostredníctvom cesty vyznačenej paketom. To umožňuje zriadenie plne duplexnej cesty. K zníženiu réžie hľadania novej cesty je query paket zahodený, ak počas zaplavovania (flooding) narazí na uzol, ktorý už pozná cestu k cieľovému uzlu. Potom, čo bola cesta zriadená, sa udržiava tak dlho, kým ju zdroj používa. Porucha linky je zdroju oznámená rekurzívne cez medziľahlé uzly a tento oznam spustí novú

sekvenciu vyhľadávania cesty.

Rozdielny prístup pre sledovanie ciest používa DSR. DSR využíva smerovanie zo zdroja, t.j. zdroj uvádza v hlavičke paketu poradie smerovacích uzlov na dátovej ceste. V DSR query paket skopíruje do svojej hlavičky identifikátory (ID) uzlov, cez ktoré prechádzal. Cieľová stanica potom načíta celú cestu z hlavičky query paketu, použije ju na odpoveď zdroju a tým poskytne zdroju celú smerovaciu cestu [3]. Dátové pakety nesú celú trasu k zdroju vo svojej hlavičke. Na minimalizáciu nákladov na objavovanie trás DSR uzol uchováva v pamäti všetky trasy, ktoré sa doposiaľ naučil. Smerovanie zo zdroja umožňuje DSR uzlom uchovanie viacerých trás k cieľu a tým oddialiť nutnosť rekonštrukcie ciest použitím alternatívnej cesty k cieľu, ak ju zdroj pozná. Ak žiadna platná cesta neexistuje, musí sa spustiť nové hľadanie ciest. Zahrnutie cesty v hlavičke paketu robí detekciu slučiek veľmi jednoduchou.

2.2.2 Hierarchické smerovacie protokoly

So zväčšovaním bezdrôtovej siete za určitý prah sa bežné ploché smerovanie stáva nemožné kvôli narastajúcej réžii na spracovanie prepojení v sieti. Jedným z riešení tohto problému a zriadením efektívnej siete je aplikovanie hierarchického smerovania. Príkladom hierarchického usporiadania siete je internetová hierarchia, ktorá je praktikovaná v káblovej sieti už dlhú dobu. Hierarchické bezdrôtové pripojenie je postavené na myšlienke organizácie sieťových uzlov do skupín a následnom pridelení rôznych funkcií uzlom v rámci aj mimo skupinu. Usporiadaním staníc do skupín sa zmenšia smerovacie tabuľky aj veľkosť aktualizačného paketu, pretože sa namiesto informácii o celej sieti posielajú údaje len o jej časti, čím sa znižuje aj zložitosť kontroly.



Obr. 2.2: Hierarchické smerovanie sietí MANET

Najbežnejší spôsob zavedenia hierarchie je vytvorenie skupín (explicit clusters) z uzlov, ktoré sú geograficky blízko seba. Každá skupina má hlavný uzol (clusterhead) na komunikáciu s ostatnými uzlami v rámci skupiny [3]. Alternatívny spôsob je zriadenie implicitnej hierarchie. Každý uzol tak má len lokálne pôsobenie a rôzne smerovacie stratégie sú použité vo vnútri a vonku pôsobiska. Komunikácia sa šíri cez prekrývajúce sa pôsobiská jednotlivých staníc. Implicitnou hierarchiou sa docieli efektívnejšie celkové smerovanie.

Keď majú mobilné uzly iba jeden všesmerový vysielač na bezdrôtovú komunikáciu, nesie toto usporiadanie označenie "logická hierarchia"pre odlíšenie od fyzickej hierarchickej siete. Podobne ako v predchádzajúcej časti nasleduje tabuľka porovnávajúca vlastnosti protokolov. Teraz sa jedná o protokoly CGSR (Cluster-Head Gateway Switch Routing Protocol), HSR (Hierarchical State Routing), ZRP (Zone Routing Protocol), a LANMAR(Landmark Routing Protocol).

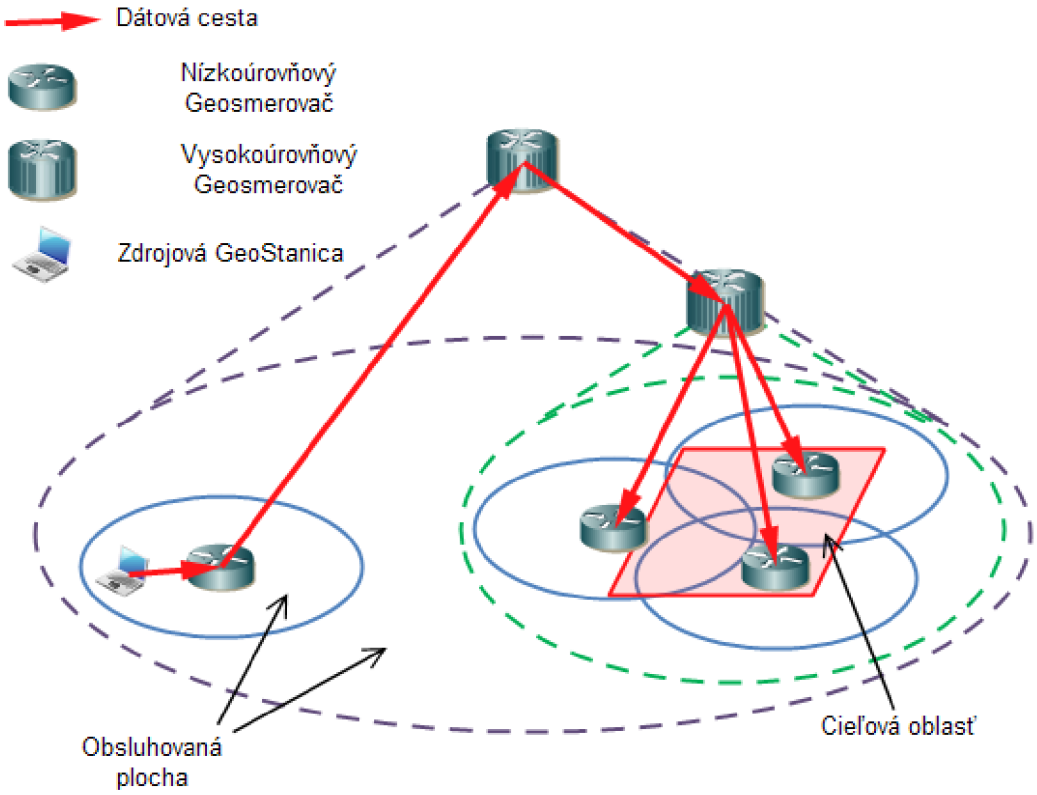
Tab. 2.2: Porovnanie vlastností hierarchických smerovacích protokolov[3]

	CGSR	**HSR**	**ZRP**	**LANMAR**
Hierarchia	explicitná, dve úrovne	explicitná, viacero úrovní	implicitná, dve úrovne	implicitná, dve úrovne
Filozofia smerovania	proaktívna, vektor vzdialenosti	proaktívna, stav linky	hybridná DV+ LS	proaktívna DV + LS
Zamedzenie slučkám	áno	áno	áno	áno
Metrika smerovania	cez kritické uzly	cez kritické uzly	lokálna najkratšia cesta	lokálna najkratšia cesta
Kritické uzly	áno (clusterhead)	áno (clusterhead)	nie	áno (landmark)
Pamäťová zložitosť	$O(N/M)$	$O(M * H)$	$O(L) + O(e)$	$O(L) + O(G)$
Výpočtová zložitosť	$O(N)$	$O(M * H)$	$O(N)$	$O(N)$

2.2.3 Smerovanie za použitia geografickej pozície

S pokrokom vo vývoji GPS (Global Positioning System) je v dnešnej dobe možné poskytovať informácie o polohe s presnosťou v ráde niekoľkých metrov. GPS poskytuje aj univerzálne časovanie. Kým informácie o polohe môžu byť použité na smerovanie k určitému miestu v distribuovaných systémoch Ad hoc, univerzálne hodiny poskytujú synchronizáciu medzi uzlami vybavenými GPS. Výskum ukázal, že

informácia o geografickej polohe uzla môže zvýšiť výkon smerovania v Ad hoc sieti. V mobilnom prostredí však treba brať do úvahy ďalší problém, ktorým je nepresnosť umiestnenia v momente, keď sa informácia o polohe použije. Všetky protokoly uvedené v tabuľke predpokladajú znalosť uzlov o svojej pozícii.



Obr. 2.3: Ukážka fungovania siete GeoCast

GeoCast umožňuje posielať správy všetkým uzlom v určitej geografickej oblasti pri použití geografických informácií namiesto logickej adresy uzla [3]. Geografická cieľová oblasť môže byť určená tromi spôsobmi: bodom, kruhom (určeným stredom a polomerom) a polygónom (určeným zoznamom bodov). Bod je reprezentovaný geografickými súradnicami. Ak je cieľom polygón alebo kruh, každý uzol v oblasti dostane správu. Geosmerovač vypočíta obsluhované oblasti ako spojenie zemepisných oblastí pokrytých sieťami, ktoré sú k nemu pripojené. Táto oblasť je potom aproximovaná jediným uzavretým polygónom. Na spísanie smerovacích tabuliek si geosmerovače vymenia údaje o obsluhovaných polygónoch. Tento prístup zostaví hierarchickú štruktúru zostavenú z geosmerovačov, pričom sa koncový užívateľ môže voľne pohybovať v sieti. Dátová komunikácia začína na stanici schopnej prijímať a odosielať geografické správy. Dátové pakety sú následne odoslané na lokálny geouzol, ktorý zodpovedá za predávanie paketov miestnemu geosmerovaču. Ten skontroluje, či

jeho obsluhovaná oblasť pokrýva cieľovú oblasť. Pokiaľ existuje časť cieľovej oblasti, ktorá nie je pokrytá daným geosmerovačom, je kópia paketu odoslaná smerovaču vyššie v hierarchii (rodičovský smerovač). Rodičovský smerovač potom skontroluje oblasti svojich podriadených smerovačov. Všetky podriadené smerovače pokrývajúce cieľovú plochu dostanú kópiu paketu. Ak oblasť smerovača spadá do zamýšľanej oblasti, paket posunie uzlom k nemu pripojeným. Multicastové skupiny sú udržiavané v geouzloch. Prichádzajúce správy sú určitý čas uložené a počas toho periodicky vysielané cez priradené multicastové adresy. Klienti sa na prijímanie správ pripoja k príslušnej multicastovej adrese. V tabuľke sa nachádzajú vlastnosti protokolov GeoCast, LAR (Location-Aided Routing), DREAM (Distance Routing Effect Algorithm for Mobility) a GPSR(Greedy Perimeter Stateless Routing).

Tab. 2.3: Vlastnosti geografických smerovacích protokolov[3]

	GeoCast	**LAR**	**DREAM**	**GPSR**
Podpora uprednostnenia lokácie	áno	áno	áno	nie
Predávanie dát na základe lokácie	áno	nie	áno	áno
Filozofia smerovania	proaktívna	reaktívna	proaktívna	proaktívna (len beacon)
Citlivý na pohyb staníc	nie	áno	nie	nie
Metrika smerovania	najkratšia cesta	najkratšia cesta	najkratšia cesta	najbližšia vzdialenosť
Zamedzenie slučiek	áno	áno	áno	nie
Najhorší scenár	neexistuje	áno (úplné zaplavovanie)	neexistuje	áno(slučky a dlhšie trasy)
Podpora viacerých prijímačov	áno	nie	nie	nie
Pamäťová zložitosť	$O(N)$	$O(N)$	$O(N)$	$O(M)$
Výpočtová zložitosť	$O(N)$	$O(e)$	$O(N)$	$O(M)$

3 IPV6

Začiatkom 90. rokov sa začalo uvažovať nad problémom vyčerpania IPv4 adries. Viacero expertov predpovedalo vyčerpanie celého adresného priestoru v priebehu pár rokov[2]. Preto sa začali práce na novom protokole IP označovanom počas vývoja ako IP Next Generation alebo IPng a v súčasnosti známom pod názvom IPv6. Tvorba nového štandardu by zabrala veľa času, preto sa hľadalo aj nejaké krátkodobé riešenie. Tým riešením je NAT (Network Address Translation – Prekladanie sieťových adries), ktoré umožňuje viacerým užívateľom používať jednu alebo zopár verejných IP adries. NAT úspešne spomalil vyčerpanie IPv4 adries a stal sa štandardnou súčasťou väčšiny sietí[2]. Preto veľa ľudí dodnes spochybňuje potrebu novej verzie IP. Široké využívanie NAT však spôsobilo zmenu otvoreného transparentného peer-to-peer Internetu na množstvo sietí klient – server.

Hoci väčšina štandardov IPv6 bola dokončená už pred rokmi, vážnejší záujem o prechod na IPv6 sa začal prejavovať len nedávno. Za tým môžeme vidieť dva hlavné dôvody. Prvým je použitie nových aplikácií využívajúcich koncept mobile IP, zabezpečenie kvality služieb, zvýšenie bezpečnosti sietí a služby typu peer-to-peer a grid computing. NAT potlačuje vývin týchto služieb a jediný spôsob ako vytlačiť NAT je zabezpečenie hojného využívania verejných IP adries. Druhým dôvodom je zrýchlenie modernizácie husto zaľudnených krajín ako Čína a India. V týchto krajinách už v súčasnosti existuje 4 a 5 úrovňová NAT hierarchia len na pokrytie súčasného dopytu[2]. IPv6 nahradí 32-bitové adresy 128-bitovými a vytvorí tak $3,4 \cdot 10^38$ IP adries, čím v dohľadnej budúcnosti vyrieši problémy s ich nedostatkom.

3.1 Adresy IPv6

128-bitobé adresy sú zapisované ako 8 16-bitových segmentov, každý segment je zapísaný v hexadecimálnej sústave medzi `0x0000` a `0xFFFF` oddelených dvojbodkou. Príkladom môže byť adresa :

```
3ffe:1944:0100:000a:0000:00bc:2500:0b0d
```

Pre jednoduchšie zapamätanie boli schválené dve pravidlá. Prvé hovorí o nepotrebnosti zápisu prvých núl v 16-bitových segmentoch, teda ak má 16-bitový segment menej ako 4 hexadecimálne hodnoty, predpokladá sa že chýbajúce číslice sú nuly na začiatku[2]. Toto pravidlo zjednoduší adresu z príkladu na:

```
3ffe:1944:100:a:0:0bc:2500:b0d
```

Veľa adries obsahuje dlhé reťazce núl, ako napríklad adresa:

```
3ffe:0000:0000:0000:0000:0000:0000:0009
```

Táto adresa môže byť skrátená na:

```
3ffe:0:0:0:0:0:0:9
```

Avšak druhé pravidlo, ktoré znie: "Ľubovoľný jeden reťazec jedného alebo viacerých segmentov zložených z núl môže byť nahradený dvomi dvojbodkami[2]."skracuje túto adresu na tvar:

```
3ffe::9
```

Dve dvojbodky v zápise smú byť použité iba raz.

Existujú tri typy IPv6 adries, unicast, anycast a multicast. Oproti IPv4 sa v IPv6 nenachádza broadcast adresa, namiesto nej je tam multicast adresa "všetkým uzlom", ktorá slúži prakticky rovnakému účelu.

Unicastová adresa je adresa, ktorá označuje jedno zariadenie. Globálna unicastová adresa je adresa ktorá je celosvetovo unikátna. Jej základný formát je na nasledujúcom obrázku.

128 bitov		
48 bitov	16 bitov	64 bitov
Globálny smerovací prefix	**ID podsiete**	**ID rozhrania**
Sieťová časť		Užívateľská časť

Obr. 3.1: Základná IPv6 unicastová adresa

Užívateľská časť adresy sa nazýva ID rozhrania. Dôvodom tohto označenia je to, že užívateľ môže mať viac ako jedno rozhranie. Jedno rozhranie môže mať viac IPv6 adries a súčasne tiež IPv4 adresu[2]. V takom prípade je ID rozhrania len jeden z viacerých identifikátorov. Dĺžka globálneho prefixu sa v určitých prípadoch môže meniť. Ak je podsietí veľmi veľa tak sa globálny prefix môže skrátiť na menej ako `/48`, ak je adresovaná len jedna podsieť tak sa môže použiť prefix `/64` a v prípade, že sa jedná len o jediné zariadenie tak dokonca prefix `/128`. Na nasledujúcom obrázku sa nachádza formát hlavičky IPv6 paketu.

Obr. 3.2: Hlavička IPv6 paketu

Verzia určuje verziu protokolu IP, v tomto prípade rovné 6, binárne zapísané ako 0110. Pole má 4 bity. Trieda prevádzky je 8-bitové pole ktoré zodpovedá poľu ToS (Type of service) pri IPv4 pakete. Využíva sa pre diferencované služby (DiffServ).

Pole značka toku je pre jedinečné pre IPv6. Účel tohto poľa je možnosť označenia určitých tokov v prevádzke, čo sú toky, ktoré nielen majú spoločný zdroj a cieľ, ale tiež patria k rovnakej aplikácii. To dovoľuje posielať rovnako označené pakety rovnakou cestou a tým zabezpečiť ich postupnosť.

Nasledujúca hlavička špecifikuje aká hlavička protokolu vyššej vrstvy nasleduje po hlavičke IPv6 paketu. Pole sa využíva k rovnakému účelu ako pole Protokol v hlavičke IPv4 paketu. Limit preskokov zodpovedá dĺžkou aj funkciou poľu TTL (Time to Live) pri verzii 4 protokolu IP. Pole určuje maximálny počet preskokov v sieti, ktoré môže paket vykonať na ceste k cieľu. Ak sa Limit preskokov rovná nule je paket zahodený.

4 OSPFV3

OSPF (Open Shortest Path First) je smerovací protokol, ktorý je určený na smerovanie IP paketov iba v rámci svojej smerovacej domény (autonómneho systému), tzn. patrí medzi Interior gateway protokoly (IGP). Na konštrukciu sieťovej topológie využíva informácie zhromaždené z ostatných dostupných smerovačov. Topológia rozhoduje o smerovacej tabuľke prezentovanej internetovej vrstve, čím umožňuje smerovanie len za použitia cieľovej IP adresy (nachádza sa v IP paketoch). OSPF podporuje siete s premennou dĺžkou sieťovej masky (VLSM – Variable-Length Subnet Mask) a beztriedne smerovanie v rámci domény (Classless Inter-Domain Routing – CIDR) [1].

Jedná sa o dynamický smerovací protokol, takže rozpoznáva zmeny v sieťovej topológii, ako napr. poruchy liniek veľmi rýchlo a konverguje na novú bezslučkovú štruktúru v intervale nikoľkých sekúnd. Na výpočet najkratších ciest sa používa metóda založená na Dijkstrovom algoritme shortest path first. Informácie o stave linky sú udržiavané na každom smerovači ako LSDB (link-state database), ktorá je stromovým grafom celej sieťovej topológie. LSBD je aktualizovaná na všetkých smerovačoch prostredníctvom pravidelného zaplavovania OSPF smerovačov [2]. Na zhotovenie smerovacej tabuľky za využíva znalosť tzv. hodnoty linky pripojenej k rozhraniu smerovača. Hodnota linky môže závisieť na rôznych faktoroch, ako napríklad vzdialenosť smerovača (round-trip time), priepustnosť linky alebo jej dostupnosť a spoľahlivosť. Hodnota linky je vyjadrená jednoduchým číslom bez jednotky. Tým sa zabezpečuje rovnomerné zaťažovanie liniek s rovnakým ohodnotením [2].

Sieť môže byť rozdelená na smerovacie oblasti (area) kvôli zjednodušeniu riadenia a optimalizácii zaťaženia. Oblasti sú označené 32-bitovými číslami, zvyčajne vyjadrenými ako dekadické zapísanie jednotlivých bajtov oddelených bodkami, rovnako ako IP adresy v IPv4. Podľa dohody oblasť 0 alebo `0.0.0.0` reprezentuje chrbticu siete (backbone). Identifikácia ostatných oblastí môže byť zvolená ľubovoľne, najčastejšie sa ako identifikátor volí IP adresa hlavného smerovača v oblasti [4]. Všetky ostatné oblasti musia mať priame alebo virtuálne spojenie s backbone oblasťou. Toto spojenie je udržiavané pomocou prepojujúceho smerovača označeného ako hraničný smerovač oblasti (Area Border Router – ABR). ABR udržiava oddelené databázy pre všetky oblasti, ktoré obsluhuje a zhromažďuje súhrn trás všetkých oblastí v sieti. OSPF nepoužíva transportné protokoly TCP/IP (Transmission Control Protocol/Internet Protocol), ale zapuzdruje sa priamo v IP datagramoch s číslom protokolu 89, čo je rozdiel oproti RIP (Routing Information Protocol) a BGP (Border Gateway Protocol). OSPF taktiež spravuje vlastnú detekciu a opravu chýb [5]. Na zaplavovanie OSPF využíva multicastové adresy. Pre siete bez multicastu sa preto v nich musí nastaviť funkcia pre vyhľadávanie susedných smerovačov. Multicast pakety ni-

kdy neprechádzajú za smerovač, teda nikdy nemajú viac ako jeden preskok. Pre multicast má podľa RFC 2328 (OSPFv2) a RFC 5340 (OSPFv3) OSPF vyhradené adresy uvedené v tabuľke.

Tab. 4.1: Multicast adresy OSPF

	IPv4	**IPv6**
AllSPFRouters	224.0.0.5	FF02::5
Designated Routers, AllDRouters	224.0.0.6	FF02::6
Všetky smerovače v oblasti	224.0.0.2	FF02::2

OSPFv3 bežiace na IPv6, na rozdiel od staršieho OSPFv2 nepodporuje autentifikáciu vnútri protokolu a namiesto nej sa spolieha na zabezpečenie sprostredkované protokolom IPv6. Ďalším rozdielom oproti OSPFv2 je výmena informácií medzi susednými uzlami, kde sú všetky uzly adresované pomocou IPv6, okrem virtuálnych spojení. Všetky informácie o prefixoch boli odstránené z LSA (link-state advertisment) a z Hello paketov, čím sa OSPFv3 stáva nezávislým od ostatných protokolov. Napriek rozšíreniu adresného priestoru na 128 bitov v IPv6, oblasti aj identifikátory smerovačov sú stále označované 32-bitovými hodnotami.

4.1 Podobnosť a rozdiely medzi OSPFv2 a OSPFv3

Medzi verziami 2 a 3 OSPF protokolu existuje pár spoločných rysov a rozdielov. OSPFv3 používa prakticky rovnaké základné typy paketov s určitými zmenami niektorých polí. Taktiež veľkosť hlavičky paketov bola zmenšená z 24 bajtov na 16 a to práve odstránením poľa autentifikácie [5]. Vytvorené bolo pole pre identifikátor inštancie. To sa používa na prácu s viacerými procesmi na jednej linke. Ak chcú dve inštancie medzi sebou komunikovať, potrebujú mať nastavené rovnaké číslo inštancie. Predvolené je 0 a pri potrebe ďalšej inštancie sa inkrementuje, samotné číslo má význam iba lokálne. Protokol samotný pracuje prostredníctvom liniek a nie pomocou podsietí ako starší OSPFv2 [5]. Mechanizmy hľadania susedných smerovačov a formovania priľahlosti ostali rovnaké. Nezmenilo sa ani zaplavovanie správami LSA a ich starnutie. Typy LSA sú tiež takmer identické. Pre protokol OSPFv3 boli však vytvorené 2 nové typy a protokol samotný zvláda aj spracovanie neznámych typov LSA. Pole volieb je z LSA hlavičky presunuté do tela správy LSA.

4.2 Druhy oblastí OSPF

OSPF domény sú rozdeľované do oblastí označených 32-bitovými identifikátormi, najčastejšie vo forme IPv4 adresy. Tým, že sa nejedná o IP adresy, nenastáva konflikt medzi identifikátormi a IP adresami v sieti. Oblasti sú logické skupiny koncových uzlov a sietí, vrátane smerovačov, ktoré majú pripojené rozhrania k ľubovoľnej zahrnutej sieti. Každá oblasť spravuje vlastnú link-state databázu (LSDB), ktorej zhrnuté informácie poskytuje zvyšku siete prepájajúci smerovač [5]. Tým, že sa informácia o topológii nešíri mimo oblasť, sa redukuje množstvo prenesených informácií medzi autonómnymi systémami. OSPF definuje viacero typov oblastí:

Oblasť backbone (označená ako oblasť 0 alebo `0.0.0.0`) predstavuje jadro siete OSPF. Všetky ostatné oblasti sú pripojené k backbone oblasti, smerovanie medzi oblasťami prebieha prostredníctvom smerovačov pripojených medzi oblasť 0 a ich vlastnú pridelenú oblasť. V štruktúre OSPF domény je oblasť 0 pripojená ku všetkým nenulovým oblastiam v rámci domény [5]. Backbone oblasť je zodpovedná za distribúciu smerovacích informácii medzi nenulovými oblasťami, preto musí susediť s ostatnými oblasťami prostredníctvom fyzickej alebo virtuálnej linky. Príkladom môže byť situácia, keď oblasť `0.0.0.1` má fyzické pripojenie k oblasti `0.0.0.0` a oblasť `0.0.0.2` toto pripojenie nemá. Má však pripojenie k oblasti `0.0.0.1`. Potom môže oblasť `0.0.0.2` použiť virtuálne spojenie cez tranzitnú oblasť `0.0.0.1` na pripojenie k backbone. Tranzitná oblasť však musí byť "priechodná", takže sa nemôže jednať o koncovú (stub) oblasť.

Koncová oblasť je pripojená k ostatným oblastiam prostredníctvom jedného spojenia, avšak neumožňuje prenos cez oblasť. Informácie, ktoré sa majú šíriť mimo koncovú oblasť sa musia šíriť jediným hraničným smerovačom (Area border router – ABR). Koncová oblasť nedostáva smerovacie správy, ktoré sú mimo autonómny systém. Hraničný smerovač sa označí ako predvolená cesta a tým redukuje veľkosť smerovacej tabuľky pre smerovače vnútri oblasti, čím zvyšuje efektivitu a rýchlosť smerovania.

Not-so-stubby oblasť (NSSA) je druh koncovej oblasti, ktorá môže importovať externé cesty autonómneho systému a odoslať ich ostatným oblastiam, avšak nedostáva externé cesty z ostatných oblastí. Priebeh smerovania je nasledovný: ASBR (Autonomous system boundary router) importuje externé cesty pomocou LSA typu 7 a konvertuje ich na LSA typu 5, ktorými potom zaplavuje ostatné oblasti. ABR sa vtedy javí pre ostatné oblasti ako ASBR.

4.3 Smerovače v OSPF

OSPF podľa funkcie definuje 4 druhy smerovačov:

- Area border router (ABR)
- Autonomous system boundary router (ASBR)
- Internal router (IR)
- Backbone router (BR

ABR – je smerovač, ktorý pripája jednu alebo viacero oblastí k backbone sieti. Je to člen všetkých oblastí, ku ktorým je pripojený. V jeho pamäti sa nachádza kópia LSDB pre každú pripojenú oblasť.

ASBR – je smerovač pripojený k viacerým smerovacím protokolom a je tým, ktorý si vymieňa smerovacie informácie so smerovačmi bežiacimi na inom smerovacom protokole. ASBR zvyčajne prevádzkuje aj exterior routing protocol napríklad BGP.

IR – vnútorný smerovač je smerovač susediaci s ostatnými smerovačmi v jednej oblasti.

BR – Backbone smerovače sú všetky smerovače, ktoré sú pripojené k backbone oblasti, nezávisle na tom, či sa jedná o ABR alebo o vnútorný smerovač backbone oblasti. ABR je stále backbone smerovač, lebo všetky oblasti musia byť pripojené k oblasti 0.

4.3.1 Designated router

Designated router (DR) posiela LSA typu 2 všetkým susedným smerovačom. Je to rozhranie smerovača, ktoré sa volí v rámci segmentu multiaccess a slúži k redukcii sieťovej prevádzky. DR je zdrojom smerovacích aktualizácií, udržuje si kompletnú tabuľku topológie a všetky ostatné smerovače s ním nadväzujú spojenie [1].

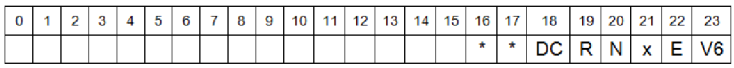
Backup Designated Router (BDR) zastáva funkciu DR, ak zlyhá pôvodný DR. BDR má druhú najvyššiu prioritu v dobe voľby DR [1].

4.4 Typy a popis paketov v OSPFv3

Všetkých 5 typov paketov v OSPFv3 obsahuje rovnakú hlavičku, v ktorej sa podľa typu paketu mení iba pole Typ [5].

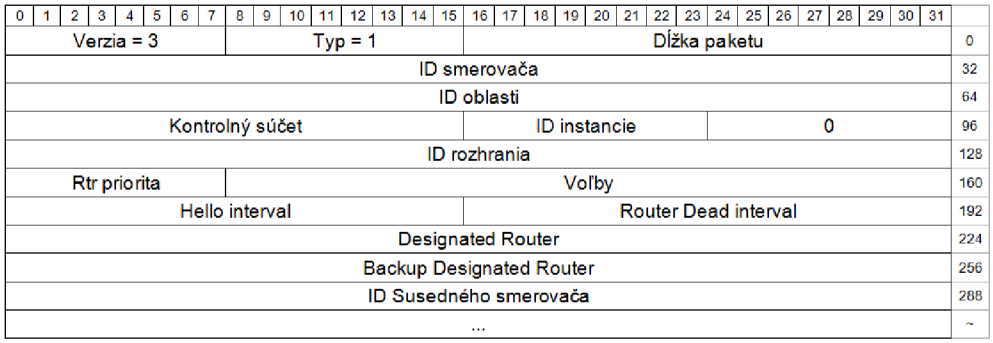
Obr. 4.1: Hlavička OSPFv3 paketu

Verzia – určuje verziu protokolu OSPF Typ – paket zodpovedá jednému z piatich typov paketov v OSPFv3 (1. hello, 2. database description, 3. link state request, 4. link state update, 5. link state acknowledgement)
Dĺžka paketu – udaná v bytoch, vrátane hlavičky
ID smerovača – ID zdroja paketu
ID oblasti – 32 bitové číslo identifikujúce oblasť, kam paket patrí, v prípade virtuálnej linky vždy 0
Kontrolný súčet – OSPFv3 používa štandardný kontrolný súčet pre IPv6 aplikácie
ID inštancie – povoľuje viacero inštancií prevádzkovať na jednej linke
0 – rezervované pole, musí ostať prázdne
Voľby – určených je 7 bitov z celkového počtu 24, jedná sa o súčasť OSPF paketov, v hello pakete sú to bajty 7 až 9



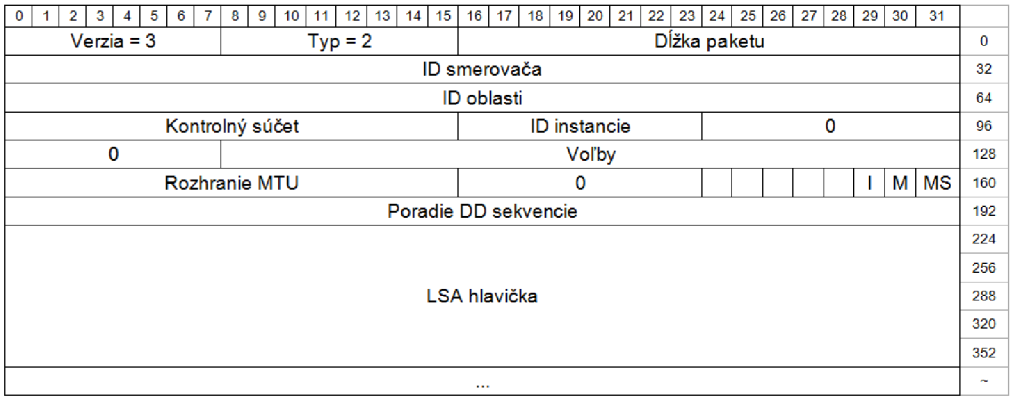
Obr. 4.2: Pole volieb

V6-bit – ak je nastavený na 0, linka je vylúčená zo smerovacích výpočtov IPv6
E-bit – určuje spôsob zaplavovania AS-external-LSA
x-bit – momentálne nepoužívaný, v minulosti použitý na MOSPF (Multicast Open Shortest Path First)
N-bit – určuje pripojenie smerovača k NSSA
R-bit – určuje, či je pôvodca aktívny smerovač
DC-bit – popisuje narábanie smerovača s demand okruhmi
*-bit – rezervované pre migráciu protokolu OSPFv2



Obr. 4.3: Hello paket

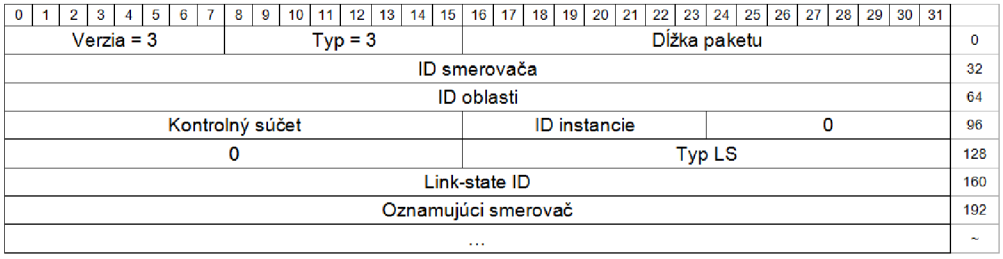
ID rozhrania – 32-bitové číslo, určuje rozhranie spomedzi všetkých rozhraní daného smerovača
Rtr priorita – používa sa pri volení DR a BDR, ak je nastavené na 0, smerovač nikdy nie je DR ani BDR
Hello interval – počet sekúnd medzi hello správami
Router dead interval – počet sekúnd, kým je smerovač prehlásený za odpojený
Designated Router ID – predpoklad identity Designated Router-a v danej sieti, nastavené na `0.0.0.0`, ak v sieti nie je Designated Router
Backup Designated Router ID – predpoklad identity BDR v danej sieti, nastavené na `0.0.0.0` ak v sieti nie je BDR
ID susedného smerovača – ID každého smerovača, ku ktorému je smerovač pripojený



Obr. 4.4: Database description paket

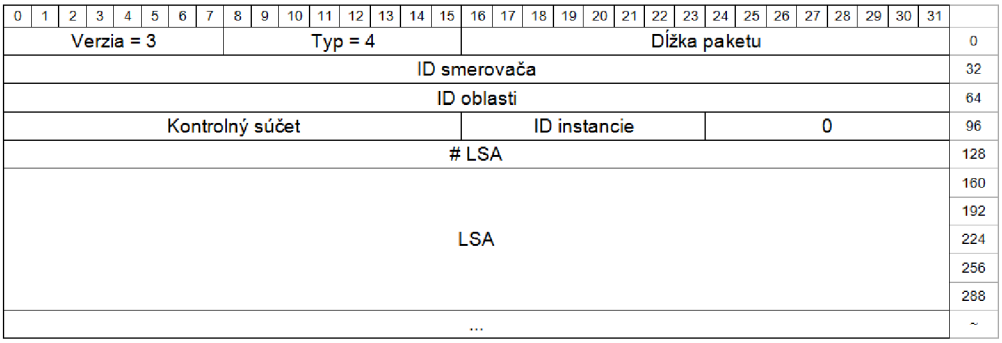
Rozhranie MTU (Maximum Transmission Unit) – maximálna veľkosť IPv6 datagramu, aký sa dá poslať z rozhrania bez fragmentácie, pri virtuálnych linkách nastavené na 0

I-bit – Init bit, ak je nastavený na 1, tak je paket prvý v sekvencii database description
M-bit – More bit, značí nasledovanie ďalších database description paketov
MS-bit – master/slave bit, ak je nastavený na 1, smerovač je master, inak slave
DD sekvencia – zoraďuje súbor database description paketov
Zvyšok paketu tvoria časti LSDB, každé LSA je popísané svojou hlavičkou.



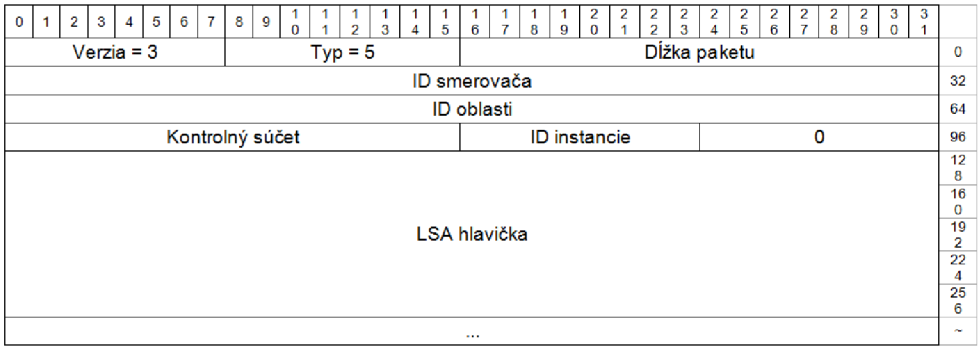
Obr. 4.5: Link state request paket

Každé vyžiadané LSA je špecifikované svojim LS typom, LS ID a oznamujúcim smerovačom. LS request je požiadavka na najaktuálnejšie LSA.



Obr. 4.6: Link state update paket

LSA – počet LSA obsiahnutých v tejto aktualizácii
Telo paketu tvorí zoznam LSA, kde každé LSA začína štandardnou 20-bytovou hlavičkou.



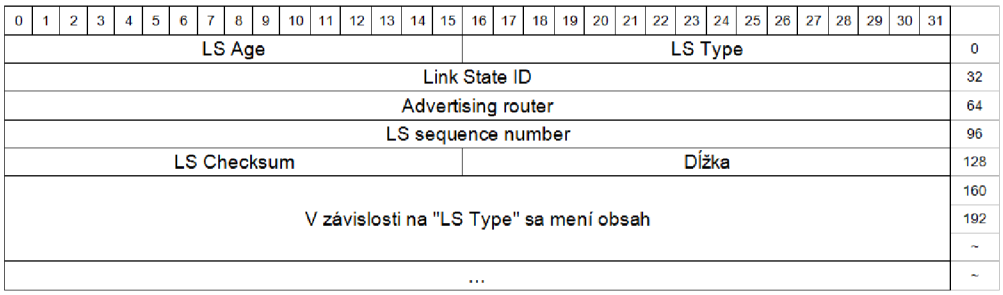
Obr. 4.7: Link state acknowledgement paket

Obsahuje všetky informácie potrebné na identifikáciu LSA aj aktuálnu inštanciu LSA.

4.4.1 Link State Advertisements - LSA v OSPFv3

Tab. 4.2: Druhy LSA

Typ	Názov
1	Router LSA
2	Network LSA
3	Inter-Area-Prefix-LSA
4	Inter-Area-Router-LSA
5	AS-External-LSA
6	nepoužité
7	NSSA-LSA
8	Link LSA
9	Intra-Area-Prefix-LSA



Obr. 4.8: Hlavička LSA paketu

LS Age – čas v sekundách, od vzniku LSA
LS Type – číslo od 1 do 4, udáva spôsob pripojenia linky (1 – bod a bod,2 – pripojenie k tranzitnej sieti,3 – pripojenie koncovej siete, 4 – virtuálna linka)
Link State ID – identifikátor LSA, ktorý bol pridelený smerovačom pri vytvorení LSA
Advertising Router – identifikátor smerovača, ktorý LSA vytvoril
LS sequence number – využíva sa na sledovanie starých alebo duplicitných LSA
LS Checksum – Fletcherov kontrolný súčet celkového obsahu LSA vrátane hlavičky [5]

5 OPNET MODELER

OPNET Modeler je simulačný program, ktorý bol vyvinutý spoločnosťou OPNET Technologies Inc. Jeho hlavnou úlohou je analýza rôznych sieťových technológií a protokolov. Základom OM je grafické prostredie, čo zrýchľuje a sprehľadňuje prácu v ňom. Taktiež umožňuje získavanie a prehľad rôznych sieťových štatistík. Táto vlastnosť dovoľuje testovanie sietí v extrémnych podmienkach a umožňuje zistenie a vyriešenie určitých problémov pred samostatným zavedením siete do prevádzky. To samozrejme vo výsledku šetrí čas a prostriedky, keďže simulácia prebieha zrýchlene. Výsledky simulácií generuje vo formáte XML (Extensible Markup Language) alebo HTML (Hypertext markup language) a dovoľuje názorné zobrazenie chovania siete pomocou animácie [7].

5.1 Zoznámenie s OPNET Modelerom

OM je tvorený hierarchickým usporiadaním editorov, ktoré spoločne popisujú spojenie štruktúr modelovej siete a protokolov. Je tvorený trojicou základných editorov:

- Project editor – editor projektu
- Node editor – editor uzla
- Process editor – editor procesu

Projektový editor graficky znázorňuje topológiu siete, spojenia medzi uzlami (Node), a taktiež prvky umožňujúce nastavenie rôznych vlastností siete, ako napríklad mobilitu uzlov, parametre vysielačov a pod. Zostavenie siete sa vykonáva metódou Chyť a pusť z palety objektov. Editor projektu môže obsahovať mapu na pozadí pre názorné zobrazenie rozloženia uzlov. Editor uzlov zobrazuje vnútornú architektúru sieťového zariadenia a vzťahy medzi funkčnými modulmi. Moduly zvyčajne predstavujú fyzické rozhrania, porty, zbernice a aplikácie alebo sieťové vrstvy. Editor procesov je rozhranie najnižšej úrovne, jeho hlavnou funkciou je tvorba konečného stavového automatu na špecifikáciu všetkých sieťových úrovní do detailu. Stavy a prechody sú graficky zobrazené a obsahujú kód v jazyku C/C++ [7].

6 PRAKTICKÁ ČASŤ PRÁCE

V tejto časti sa nachádza postup tvorby modelu OSPFv3 MANET siete v OPNET Modeleri. Kapitola je napísaná ako návod na zostavenie siete a získanie potrebných štatistík.

6.1 Návrh siete v OM a Simulácia

Postup v prvých kokoch popisuje vytvorenie projektu v OM. Potom nasleduje popis nastavenie mobilnej domény a hlavne zostavenia sieťovej topológie na pracovnej ploche. Ako ďalšia časť nasleduje postup nastavenia rozhraní jednotlivých smerovačov v topológii a výber štatistík, ktoré chceme sledovať. Ako posledný krok sa vykonáva nastavenie globálnych parametrov, ako je nastavenie adries a smerovacích protokolov v sieti a taktiež vlastností všetkých vysielačov na pracovnej ploche.

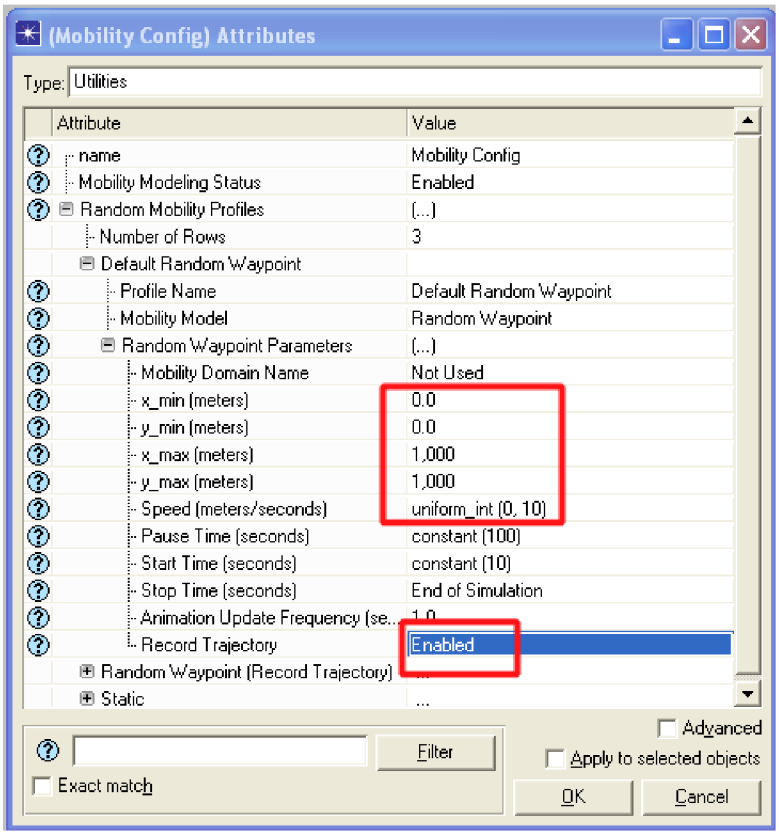
6.1.1 Vytvorenie projektu

1. Spustíme OPNET Modeler.
2. Z menu vyberieme položku **File** – **New** a potvrdíme **Project**.
3. Zadáme názov projektu (napr. OSPFv3) a scenára (napr. scenario1).
4. V **Startup Wizard** zadáme **Create empty scenario**.
5. Z dostupných možností rozlohy vyberieme **Campus**, položku **Use metric units** necháme zaškrtnutú.
6. Rozmery pracovnej plochy nastavíme na **1 x 1 kilometers**.
7. V nasledujúcom menu zvolíme predvolenú paletu objektov, v našom prípade **MANET** a pole vedľa prepneme z **No** na **Yes**.
8. Skontrolujeme nami definované nastavenia a potvrdíme stanovené voľby kliknutím na **Finish**.[10]

6.1.2 Vytvorenie sieťovej topológie a nastavenie uzlov

9. Po kliknutí na **Finish** vidíme nami stanovenú pracovnú plochu a **Object Palette** s rozbalenou položkou **MANET**. **Object Palette** slúži na pridávanie objektov na plochu.
10. Z **Object Pallete** vyberieme položku **wlan_ethernet_router** a metódou **Chyť a pusť** umiestnime potrebný počet na pracovnú plochu (napr. 10).
11. Z **Object Palette** zvolíme **Mobility Domain** a natiahneme ju cez celú pracovnú plochu. To stanovuje uzlom kde sa smú pohybovať jednotlivé stanice. Na ploche môže byť nastavených aj viacero mobilných domén.

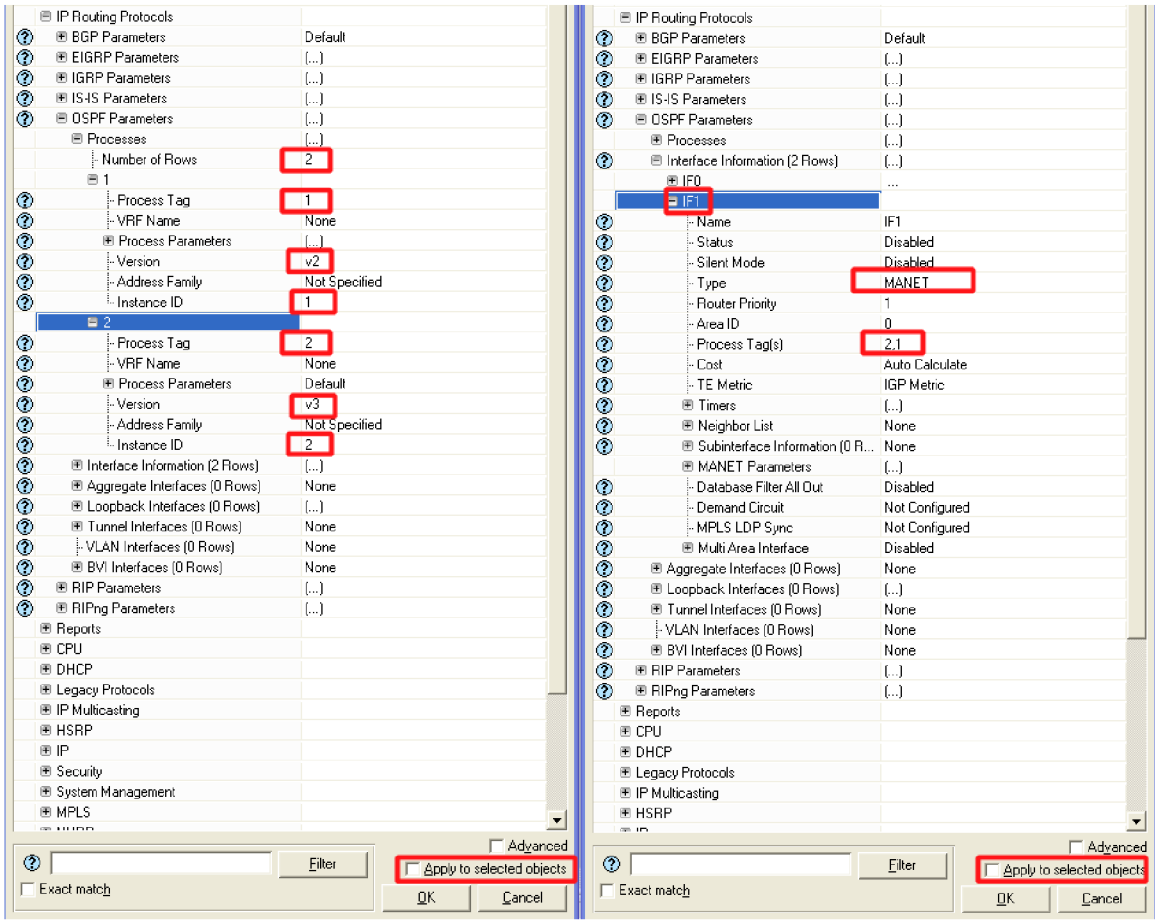
12. Označíme **všetky uzly** v sieti a v **Hlavnom menu** klikneme na **Topology** – **Random Mobility** – **Set Mobility Profile** a vyberieme **Default Random Waypoint**. Tento krok vytvorí na ploche **Mobility Config**.
13. Klikneme RMB (pravým tlačítkom myši) na **Mobility Config** a rozklikneme položky **Random Mobility Profiles** – **Default Random Waypoint** – **Random Waypoint Parameters** a nastavíme zhodne **x_max (meters)** a **y_max (meters)** na hodnotu **1000** (kvôli nášmu nastaveniu rozmerov Campus pracovnej plochy), prípadne ešte nastavíme rozsah rýchlosti pohybu uzlov **Speed (meters/second)**. Prepneme položku **Record Trajectory** na **enable**. Týmto krokom nastavíme náhodnú mobilitu, jej maximálne medze a rýchlosť pohybu uzlov po ploche. Posledná položka povolí zaznamenanie trajektórie, pre prípadnú animáciu.



Obr. 6.1: Mobility config.

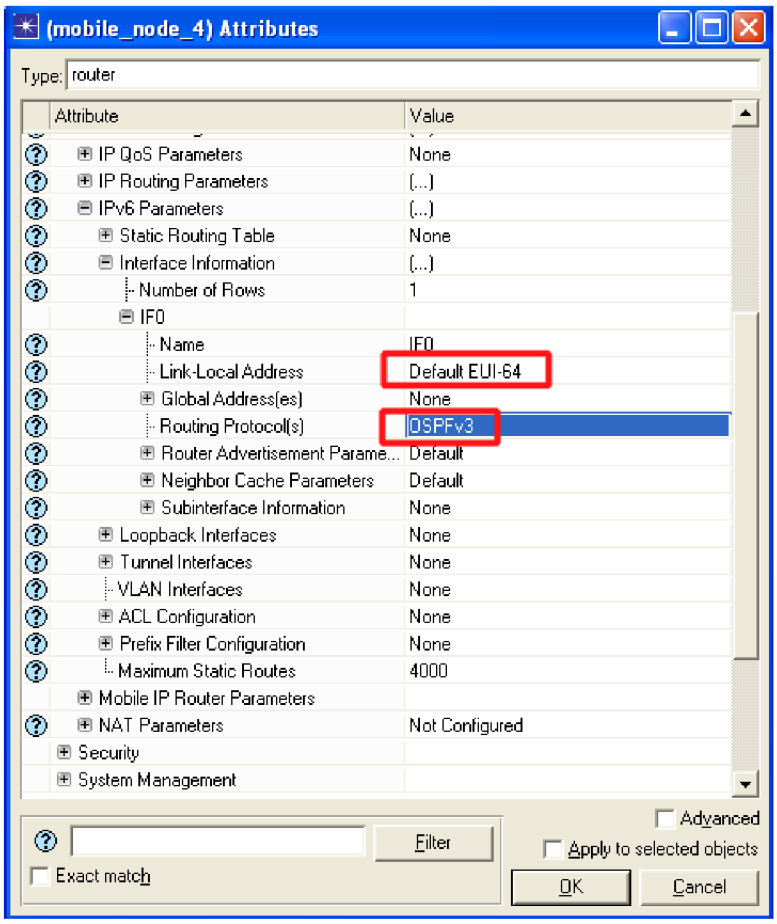
14. Označíme všetky sieťové uzly (**wlan_ethernet_router**) napríklad kliknutím RMB a vybraním **Select Similar Nodes**. Opäť kliknúť RMB na uzol a vybrať **Edit Attributes**. Tento postup označí všetky prvky jednej skupiny, čo je vhodné najmä ak na nich chceme nastaviť rovnaké parametre.

15. Postupne sa preklikáme cez položky **IP Routing Protocols – OSPFv3 Parameters – Processes** kde zmeníme hodnotu **Instance ID** na **2**. Zaškrtnúť **Apply to selected objects** a zmeny potvrdíme **OK**. Tu sa nastavujú procesy smerovacích protokolov.
16. V ďalšom kroku volíme z **Edit Attributes – IP Routing Protocols – OSPF Parameters – Interface Information**, kde **IF0** ponecháme bez zmeny nastavujeme len **IF1 – Type – MANET** a priradíme obe Process Tagy (klik na číslo **1** a v tabuľke u oboch zvoliť **Yes**) Nezabudnúť na zaškrtnutie **Apply to selected objects** a zmeny potvrdíme **OK**. Bod 16. popisuje nastavenie rozhrania smerovača na smerovací protokol OSPFv3.



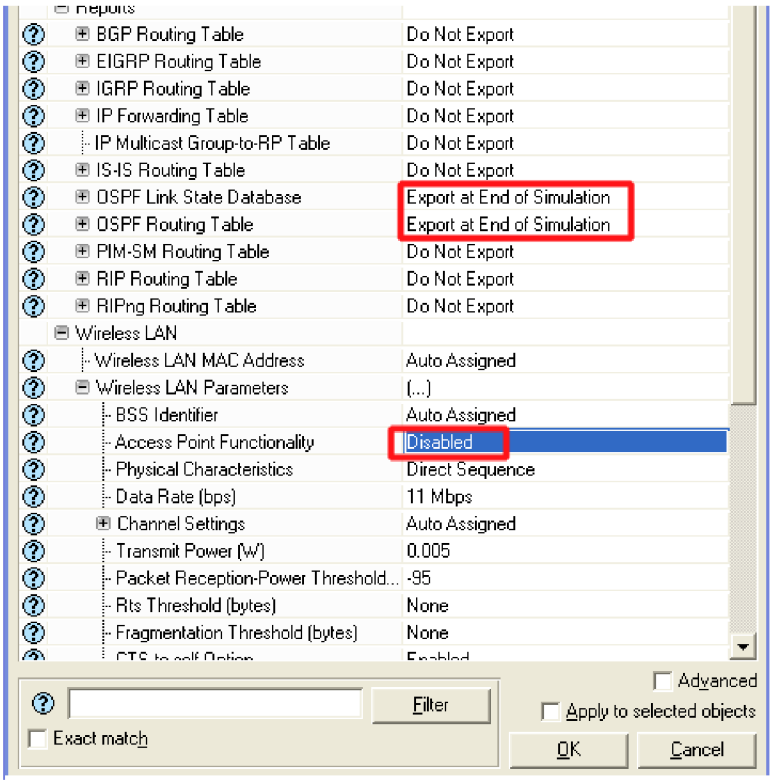
Obr. 6.2: Nastavenie smerovačov

17. Ďalej nastavujeme **Edit Attributes – IP – Ipv6 Parameters – Interface Information – Number of Rows – 1** a vo vytvorenom **IF0** nastavíme **Link-Local Address** na **Default EUI-64** a **Routing Protokol(s)** na **OSPFv3** (ostatné vypnúť). Nastavenie smerovacieho protokolu pre IPv6 na OSPFv3 a predvolených adries liniek.



Obr. 6.3: Nastavenie rozhraní

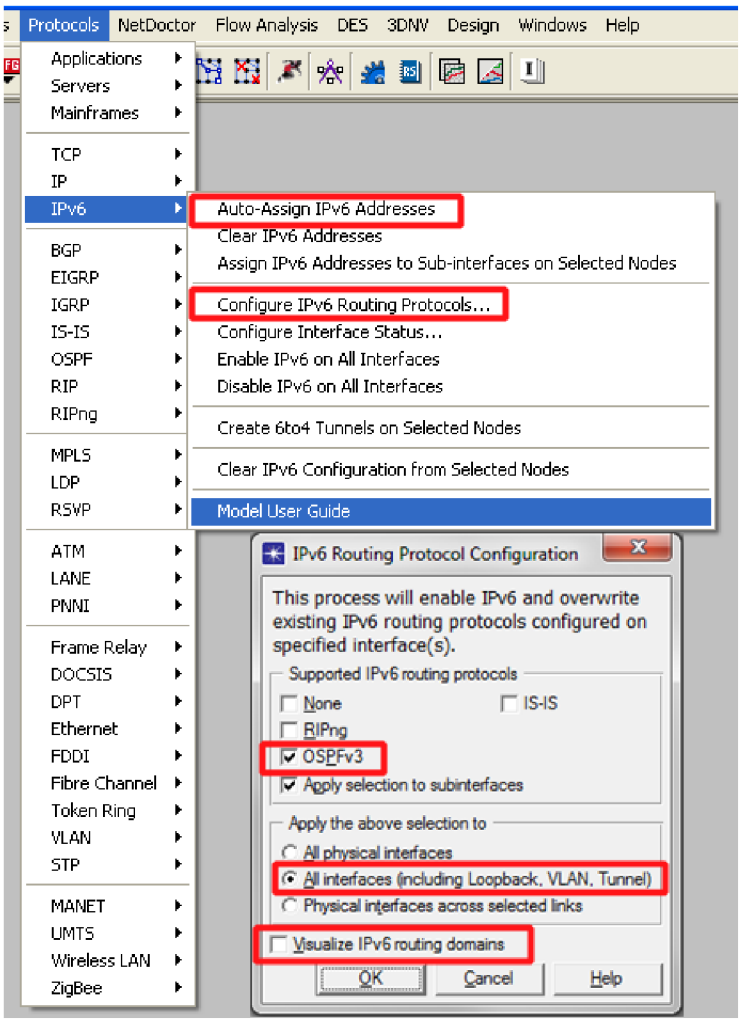
18. Export smerovacích a LSDB tabuliek nastavíme Zmenou položiek **Edit Attributes – Reports – OSPF Link State Database** a **OSPF Routing Table** na **Export at End of Simulation**. Zmeníme ešte položku **Wireless LAN – Wireless LAN Parameters – Access Point Functionality** z **Enabled** na **Disabled**. Tento krok sa vykonáva preto, aby sa informácie v sieti prenášali prostredníctvom smerovačov a nie prostredníctvom prístupových bodov. Zaškrtnút **Apply to Selected objects** a potvrdiť stlačením **OK**.



Obr. 6.4: Nastavenie exportu smerovacích tabuliek

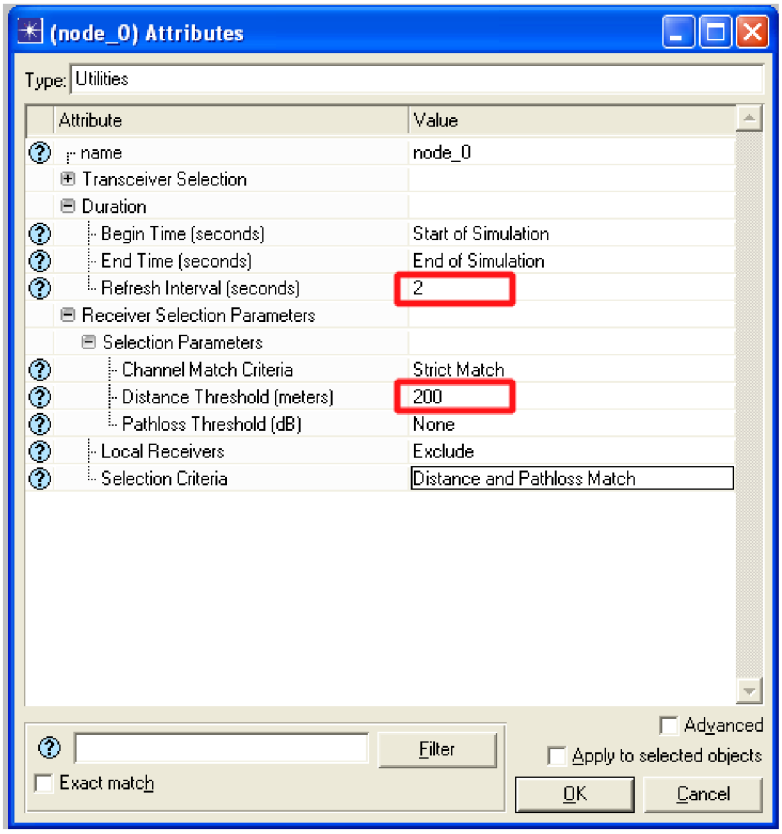
6.1.3 Vytvorenie prevádzky a nastavenie globálnych parametrov

19. Z **Object Palette** vyberieme **IP_711_Voice** (vyhľadáme pomocou vrchného riadku) a nastavíme ho na 2 uzly v sieti. V jeho atribútoch (RMB a **Edit Attributes**) nastavíme **Traffic Start Time** napr. na **2 minutes** a **Traffic Mix** na **0,1%** Explicit. Týmto postupom sa zabezpečí aby stanice nazačali vysielať dáta veľmi skoro, kým ešte nie je sú v sieti pripravené smerovacie cesty a smerovače si nezapíšu potrebné dáta do smerovacích tabuliek.
20. Vyberieme štatistiky, ktoré chceme sledovať u dátovej prevádzky pomocou RMB a **Choose Individual DES Statistics**. Môžeme vybrať všetky.
21. Nastavíme globálne parametre ako sú smerovací protokol a automatické pridelenie IPv6 adries ku rozhraniam smerovačov. V hlavnom menu položka **Protokols – Ipv6 – Auto Assign IPv6 addresses** a potom **Configure IPv6 Routing Protocols ...** v tabuľke vybrať **OSPFv3** a ostatné odznačiť, prepnúť radio button na **All interfaces (including Loopback, VLAN, Tunnel)**, taktiež vypnúť **Visualize IPv6 routing domains**.



Obr. 6.5: Nastavenie globálnych parametrov

22. Ďalším krokom je nastavenie dosahu bezdrôtových rozhraní smerovačov a periódu, s ktorou sa má tento parameter sledovať. Z **Object Palette** vyberieme prvok **Rxgroup Config** a umiestnime ho na plochu. Následne naň klikneme RMB a volíme **Edit Attributes – Reciever Selection Parameters – Selection Parameters – Distance Treshold (meters)** a nastavíme na **200**. a **Duration – Refresh Interval (seconds)** volíme **2**.



Obr. 6.6: Nastavenie Rxgroup

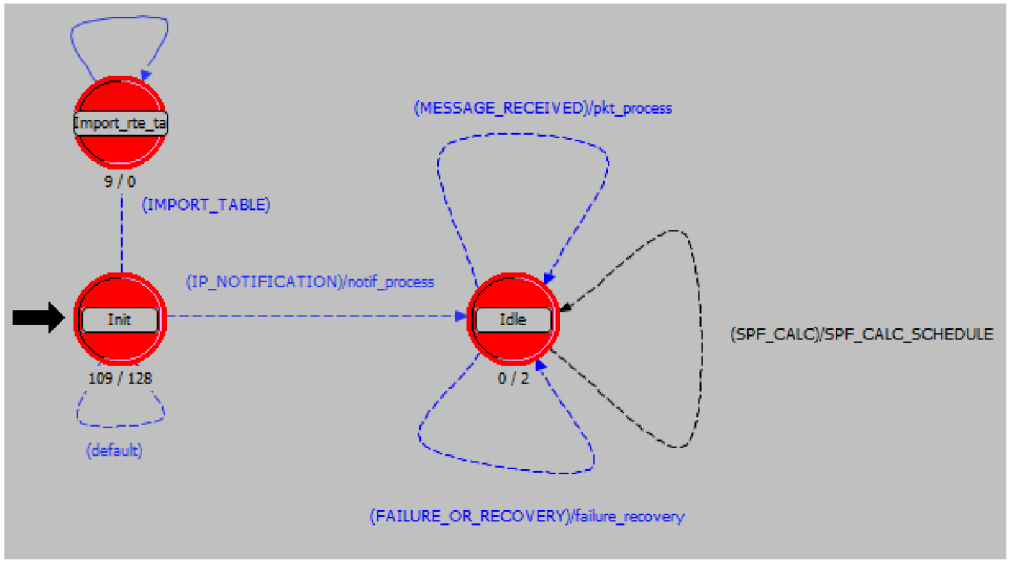
23. Vyberieme DES štatistiky: RMB na pracovnú plochu a klikneme na **Choose Individual DES Statistics** a zvolíme štatistiky **OSPF** pri oboch **Global** aj **Node Statistics**. Ďalšie vybrané štatistiky sú **Node Statistics – IP – Traffic Dropped (packets/sec)**, **Traffic Recieved (packets/sec)** a **Traffic Sent (packets/sec)**.
24. Posledné nastavenie pred spustením simulácie je: z hlavného menu vyberieme **DES – Configure/Run Discrete Event Simulation (Ctrl+R)** a v otvorenom okne volíme **Inputs – Global Attributes – IP – Ipv6 Interface Address Export – Enable** a **IP Routing Table Export/Import** na **Export** a **Inputs – Global Attributes – Simulation Efficiency – OSPF Sim Efficiency – Disable**. Tento parameter nastaví odosielanie smerovacích informácii aj po uplynutí času 260 sekúnd, inak by simulácia predpokladala, že sa už topológia siete nemení a pracovala by s neaktuálnymi informáciami získanými za tento krátky okamih.
25. Teraz je projekt pripravený na simuláciu, ktorú spustíme tlačidlom **Run**.

6.2 Procesný model OSPFv3

Procesor obsluhujúci smerovací protokol OSPFv3 sa nachádza okrem iných aj v modeloch `wlan2_router` a `wlan_ethernet_router` z objektovej palety MANET. O obsluhu OSPFv3 sa stará procesor ospf, ktorý okrem toho pracuje aj s protokolom OSPFv2.

6.2.1 Procesor ospf

Po otvorení procesoru ospf sa zobrazí jeho procesný model. Procesný model ospf obsahuje 3 nevynútené stavy, menovite **Import_rte_table**, **Init** a **Idle**. Do stavu **Import_rte_table** sa dá dostat len v prípade, že je atribút „I“P Routing Table Export/Import„n“astavený na Import. Do stavu **Init** sa dá dostať dvomi cestami, buď získaním prerušenia začiatku simulácie alebo prerušením poruchy/zotavenia. Vo vstupnej časti tohto stavu sa vykonáva len inicializácia pre prerušenie z dôvodu začiatku simulácie. Postupne sa inicializujú globálne premenné, kontrolujú sa parametre animácie, proces sa registruje v model-wide registri a registrujú sa štatistiky pre OSPF pakety a konvergenciu siete. Vo výstupnej časti sa pokračuje v inicializácii procesu OSPF. Ak boli získane nejaké pakety OSPF pokiaľ protokol nebol nastavený tak tie pakety sa v tejto chvíli zničia pomocou funkcie *ospf_msg_destroy*, pretože ich zatiaľ uzol nedokáže spracovať. Vo výstupnej časti stavu **Init** sa vytvára ID oblasti ktorá bude reprezentovať chrbticu, nastavia sa predvolené IP adresy pre OSPF multicast a vytvorí sa štruktúra smerovačov a smerovacie tabuľky. Posledný stav je stav **Idle**, do ktorého sa dostane proces po inicializácii.



Obr. 6.7: Procesný model OSPF

HB procesného modelu ospf obsahuje hlavičkové súbory, deklarácie funkcií, štruktúry a premenné možných odoberaných štatistík. Medzi hlavičkovými súbormi sa nachádzajú napríklad prg_djk.h, ktorý obsahuje Dijkstrov algoritmus na výpočet najkratšej cesty a tiež hlavičkové súbory špecifické pre OSPF ako sú ospf_lsa.h, ospf_rte_table.h, ospf_neighbor.h, ospf_area.h, ospf_v3_support.h a podobne.

Výpis funkcií

ospf_pkt_to_process_forward – Predáva paket dcérskemu procesu.

ospf_fail_recover – Vyvolanie funkcie na obnovenie alebo zlyhanie rozhrania.

ospf_ip_discover – Inicializuje premenné na import smerovacej tabuľky.

ospf_sv_init – Inicializácia stavových premenných OSPF procesu, môže podporovať viac rozhraní OSPF (jedno OSPFv2 a ľubovoľný počet OSPFv3).

ospf_notif_process – V prípade dostania IP_NOTIFICATION pokračuje vo zvyšku inicializácie.

ospf_msg_destroy – Zničenie správy a jej ICI pomocou funkcii OM (op_pk_destroy a op_ici_destroy).

ospf_redist_init – Nastaví stavové premenné spojené s prerozdelením (redistribúciou).

ospf_redist_set – Vytvorí maticu redistribuovaných smerovacích procesov podľa IP adresy.

ospf_external_route_info_process – Konverzia externej cesty do zodpovedajúceho LSA a jej pridanie do zoznamu v smerovači.

ospf_calculate_router_id – Má 3 použitia : 1 – zistenie či užívateľ nastavil router ID, 2 – výpis chyby ak užívateľ router ID nenastavil, 3 – aspoň jedno rozhranie na smerovači má nastavené router ID a dané router ID nastaví na ostatných rozhraniach

ospf_intf_info_read – Funkcia na čítanie informácii v riadku Interface Information alebo Subinterface Information spojených s parametrami OSPF.

ospf_router_area_config – Získanie ceny spoja k pripojeným oblastiam a vytvorenie zoznamu oblastí.

ospf_router_area_summary_config – Konfiguruje rozsah adries oblastí v zozname oblastí založenom na celkovom zozname oblastí.

ospf_virtual_link_config – Tvorba virtuálnej linky, smerovač musí byť ABR (area boarder router), musí mať aspoň jedno rozhranie pripojené k tranzitnej oblasti a tou nesmie byt chrbtica (backbone 0.0.0.0).

ospf_area_list_create – Vytvorí zoznam oblastí udaných užívateľom a dátové štruktúry prináležiace oblastiam.

ospf_loopback_intf_config – Konfiguruje loopback rozhrania v každej oblasti.

ospf_loopback_intf_create – Vytvorí loopback rozhranie za zaradí ho do zoznamu loopback rozhraní.

ospf_router_create – Vytvorí a alokuje dátovú štruktúru OSPF smerovača.

ospf_connected_link_fail_recover_check – Kontrola či je objekt, ktorý sa obnovil/zlyhal pripojený k aktívnemu OSPF rozhraniu na danom uzle a ak áno tak vyvolať zodpovedajúcu reakciu na daný stav (udalosť ako Interface_Down/Interface_Up).

ospf_lsdb_export_ot – Export základnej smerovacej tabuľky ako správu v záznamoch.

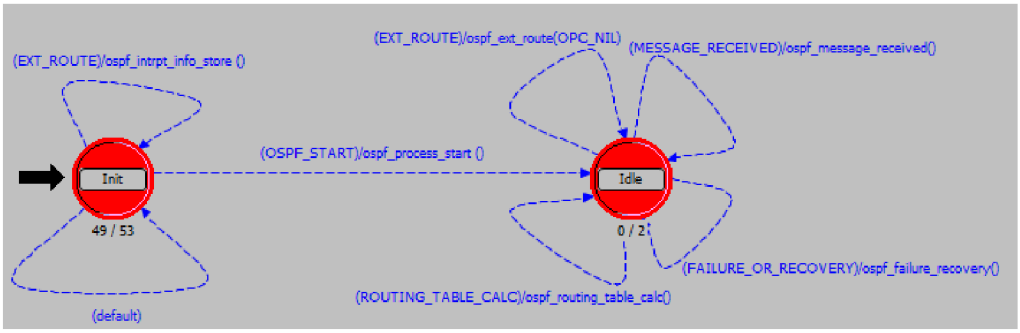
ospf_rte_table_to_ot_export – Export smerovacej tabuľky ako OT reportu .

ospf_default_route_information_config_read – Číta informácie týkajúce sa predvolenej cesty.

ospf_process_is_running_on_intf – Stanoví či je proces nastavený pre beh na danom rozhraní.

6.2.2 Dcérsky proces ospf_process

Procesný model OSFP obsahuje jeden dcérsky proces (child process) s názvom ospf_process, ktorý sa skladá z dvoch stavov, menovite **Init** a **Idle**. Stav **Init** sa opäť stará o inicializáciu procesu OSPF. V vstupnej časti tohto stavu sa naplánuje čas prerušenia pre štart protokolu OSPF. Vo výstupnej časti stavu **Init** je ošetrené prerušenie, ak prišlo skôr, ako bolo naplánované vo vstupnej časti procesu a taktiež spustenie OSPF procesu, ak počas vykonávania tejto obsluhy čas pokročí za pôvodný plán prerušenia z vstupnej časti. Spustením OSPF procesu (pomocou OSPF_START) sa proces dostane do stavu **Idle**.



Obr. 6.8: Stavový automat ospf_process

V HB sa opäť nachádzajú hlavičkové súbory a definície funkcií. Hlavičkové súbory sú až na dve výnimky zhodné s tými ktoré sú v HB rodičovského procesu. Medzi deklarovanými funkciami sú funkcie:

ospf_process_start – Spúšťa beh OSPF.

ospf_start_processing – Vygeneruje udalosť Interface_Up pre všetky OSPF rozhrania na tomto smerovači. To spustí spracovanie OSPF na všetkých aktívnych OSPF rozhraniach.

ospf_stop_processing – Zastaví spracovanie OSPF na tomto uzle, čo znamená, že nastaví všetky OSPF rozhrania na vypnuté, vyčistí smerovacie tabuľky a LS databázu.

ospf_root_message_dispatch – Spracuje a odošle OSPF správu príslušnému dcérskemu procesu.

ospf_router_interface_find – Nájde a vráti rozhranie na tomto smerovači, ktoré zodpovedá adrese, inak vráti OPC_NIL.

ospf_hello_message_rcvd – Spracuje hello správu, táto procedúra zodpovedá RFC 1247.

ospf_dbase_desc_message_rcvd – Spracovanie DD (database description) správy.

ospf_neighbor_dbase_sum_free – Procedúra na uvoľnenie posledného database summary LSA.

ospf_link_state_request_message_rcvd – Spracúva prichádzajúce Link State Request správy.

ospf_link_state_update_message_rcvd – Spracúva prichádzajúce Link State Update správy.

ospf_link_state_ack_message_rcvd – Spracúva prichádzajúce Link State Acknowledgenent správy.

ospf_rcvd_pkt_header_check – Funkcia na spracovanie hlavičky prichádzajúcich OSPF paketov, 1. – overí Správnosť Router ID a Area ID, 2. – overí, či je paket z priamo pripojeného smerovača alebo pripojeného prostredníctvom virtuálnej linky a vráti ukazateľ na rozhranie. Ak v jednom bode zlyhá, tak vráti OPC_NIL.

ospf_connected_link_fail_recover_check – Skontroluje, či je obnovujúci/zlyhávajúci objekt linka pripojená k aktívnemu OSPF rozhraniu. Ak áno, potom vyvolá zodpovedajúcu udalosť Interface_Down/Interface_Up.

ospf_redist_get_best_route – Vráti LSA obsahujúce informácie o najlepšej ceste z IP tabuľky pre danú sieť. Tiež vráti smerovací protokol najlepšej cesty v ukazateli ext_rte_proto.

ospf_sim_time_compare_proc – Porovnáva vstupné argumenty pre zachovanie chronologického usporiadania záznamov v hlavnej smerovacej tabuľke.

ospf_empty_lsa_garbage_collector – Vyprázdňuje LSA garbage collector, ale pred zničením LSA správ zaisťuje, že na LSA nie sú žiadne referencie v LSA zoznamoch.

ospf_pkt_header_mismatch_log – Z rozhrania, na ktoré paket prišiel a susedného smerovača, ktorý paket odoslal, napíše správu do záznamu.

ospf_asbr_announcement_originate – Táto funkcia sa používa na oznámenie, že sa jedná o ASBR (Autonomous System Boundary Router).

ospf_default_route_lsa_originate – Funkcia na oznámenie predvolenej cesty ostatným OSPF smerovačom. Vytvorí externé LSA a zaplaví mím sieť.

ospf_default_route_lsa_create – Funkcia, ktorá vytvára predvolenú cestu 0.0.0.0. Využíva sa, ak je pre proces povolený parameter Default Route Originate.

ospf_hello_message_process_MANET – Spracovanie hello správ z MANET rozhrania.

ospf_hello_seq_within_window – Funkcia na stanovenie, či current_seq je v oblasti window_size.

ospf_find_router_in_rcvd_router_state – Funkcia na vyhľadanie ID smerovača v zozname, za predpokladu, že zoznam obsahuje všetky ID smerovačov v sieti vo vzostupnom poradí, v podstate sa jedná o binárne vyhľadávanie.

ospf_intrpt_info_store – Získa odkaz na ICI ktoré priviedlo proces do daného stavu.

ospf_ext_route – Tento stav je spustený, ak smerovač dostane vzdialené prerušenie. Stav zapúzdruje externé smerovacie informácie, ktoré boli získané smerovačom, pričom sa jedná o iný IGP (IGRP alebo RIP), ktorý beží na smerovači.

ospf_message_received – Stará sa o prichádzajúce OSPF správy a získava adresu rozhrania, na ktoré správa prisla.

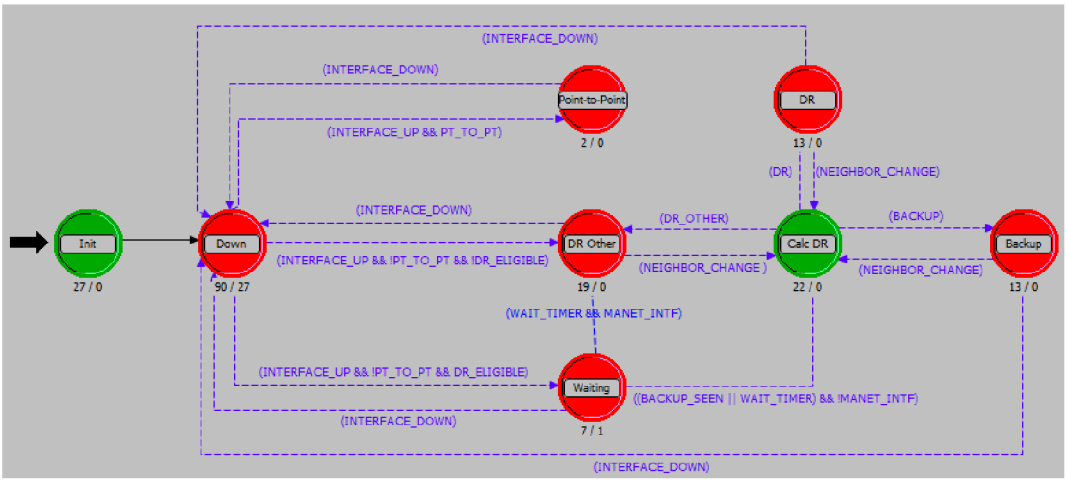
ospf_routing_table_calc – Spúšťa prepočet smerovacej tabuľky.

ospf_failure_recovery – Spúšťa prerušenia týkajúce sa obnovy/zlyhania na tomto uzle.

6.2.3 Dcérsky proces ospf_interface_v2

Ďalším dcérskym procesom je proces ospf_interface_v2, ktorý má na starosti voľbu DR rozhrania smerovača (designated router). Proces sa začína stavom **Init**, v ktorom sa inicializujú stavové premenné a rozhoduje sa či sa jedná o OSPFv2 alebo OSPFv3 a podľa toho sa registrujú multicastové adresy (IPv4 pre OSPFv2 a IPv6 pre OSPFv3). Zo stavu **Init** sa po vykonaní príkazov dostáva rozhranie do stavu **Down**, ktorého vstupná časť sa obsluhuje len v prípade, že sa do daného procesu dostane z iného stavu ako **Init**, inak sa automaticky prechádza na výstupnú časť. Vstupná časť sa vykonáva len v prípade, že sa do stavu **Down** dostane proces zo stavu **DR_Other**. V tejto časti sa resetujú všetky stavové premenné a časovače na odosielanie správ sa zrušia. Vo výstupnej časti stavu **Down** sa začnú odosielať správy HELLO, v prípade MANET rozhrania sa ešte inicializujú dynamické premenné. Zo stavu **Down** sa môže proces dostať do viacerých stavov, podľa toho či sa jedná o rozhranie point-to-point, alebo nie a v tom prípade, či je rozhranie schopné pracovať ako designated router (DR). V prípade rozhrania point-to-point sa prechádza do stavu Point-to-Point, kde proces zotrvá kým nenastane zmena jeho

stavu, vtedy sa vráti do stavu **Down**. Ak rozhranie nedokáže pracovať ako DR, tak prechádza do stavu **DR_Other**. V tomto stave sa zmení jeho rozhranie na OspfC_Interface_State_DR_Other. Zo stavu **DR_Other** sa rozhranie môže dostať dvomi cestami. Buď sa rozhranie vypne a dostane sa do stavu **Down**, alebo nastane zmena jeho susedného smerovača a dostane sa do stavu **Calc_DR**, kde sa pomocou funkcie ospf_interface_dr_elect alebo ospf_interface_mdr_elect (MDR v prípade MANET siete) prepočíta či sa jedná o DR (následne prechádza do stavu **DR**) alebo Backup DR (následne prechádza do stavu **BDR**). Ak sa nejedná ani o jeden z týchto prípadov, tak proces sa vracia do stavu **DR_other**. Zo stavov **DR** a **BDR** sa proces dostane v prípade, že sa rozhranie vypne (prechádza do stavu **Down**) alebo nastane zmena susedného smerovača (vtedy nasleduje návrat do stavu **Calc_DR**). V prípade, že sa po inicializácii a po opustení stavu **Down** zistí, že sa jedná o rozhranie, ktoré dokáže pracovať ako DR, tak prechádza do stavu **Waiting**, kde rozhranie zmení svoj stav na OspfC_Interface_State_Waiting a následne prechádza do stavu **Calc_DR**.



Obr. 6.9: Stavový automat ospf_interface

V HB sa nachádza deklarácia funkcií, ktoré pracujú so zmenami stavu rozhrania smerovača ktorý pracuje na protokole OSPF (*ospf_interface_state_change*), funkcie na voľbu DR a BDR rozhraní (*ospf_interface_mdr_elect*) a tiež funkcia na odosielanie MANET HELLO správ (*ospf_interface_mdr_hello_send*). Celkový výpis funkcií:

ospf_interface_sv_init – Inicializuje stavové premenné pre tento proces. Hodnoty pre tieto premenné sa získavajú z atribútov rozhrania. Funkcia ma tiež za

úlohu zaplniť komponenty v dátovej štruktúre rozhrania a registráciu dynamického stavu ako časť procesného registra.

ospf_interface_hello_timer_set – Spustí časovač hello správ na danom rozhraní a naplánuje ďalšie odosielanie hello správ.

ospf_interface_current_event – Vráti udalosť, s ktorou bol proces rozhrania vyvolaný alebo naplánovaný.

ospf_interface_state_change – Zmení prevádzkový stav procesu rozhrania. Funkcia by mala byť volaná na začiatku každého OPNET stavu ktorý súvisí s OSPF prevádzkovým stavom.

ospf_interface_wait_timer_set – Nastaví časovač pre rozhranie v stave WAITING. Keď časovač vyprší, stavový diagram prejde do stavu Calc DR.

ospf_interface_dr_elect – Vykonáva voľbu DR pre sieť, ku ktorej je smerovač pripojený. Stavové premenné pre DR a BDR sú touto procedúrou modifikované.

ospf_interface_dr_choose – Vyberie DR z obdŕžaného zoznamu, podľa stavu choose_backup_dr sa rozhoduje medzi výberom algoritmu pre DR alebo BDR, vráti ukazovateľ na susedný smerovač vybraný ako (B)DR. Môže vrátiť aj OPC_NIL ak je zoznam prázdny alebo ak žiaden zo susedov sa nedeklaruje ako DR.

ospf_interface_info_read – Funkcia na čítanie informácií uložených v riadku Interface Information alebo Subinterface Information.

ospf_interface_mdr_elect – Algoritmus výberu MDR (MANET DR), každé rozhranie stanoví svoj MDR status.

ospf_interface_determine_if_MDR – Funkcia na stanovenie či je rozhranie MDR.

ospf_interface_determine_if_BMDR – Funkcia na stanovenie či je rozhranie BMDR.

ospf_interface_mdr_hello_send – Funkcia na vyslanie hello správy hneď po MDR výpočte.

6.2.4 Dcérsky proces ospf_neighbor_v2

Posledný dcérsky proces sa nazýva ospf_neighbor_v2. Tento stavový automat sa zaoberá susednými a priľahlými smerovačmi. Proces začína v stave **Init**, v ktorom inicializuje stavové premenné a následne prechádza do stavu **Down**. Do stavu **Down** sa proces môže dostať prakticky z každého stavu v stavovom modeli. V prípade, že sa do tohto stavu dostane inak ako zo stavu **Init**, tak zresetuje stavové premenné a vynuluje časovače. Zo stavu Down sa môže proces dostať dvomi možnými spôsobmi. Buď získaním HELLO paketu alebo zistením, že susedný smerovač opäť odpovedá. Zo stavu **Down** sa presúva do stavu **Init**. Vo vstupnej časti **Init** procesu vymaže smerovač svoju LSA databázu a prechádza do výstupnej časti, kde získa súčasný stav susedného smerovača. Ak získa použitím funkcie *ospf_neighbor_current_event*

stav TWO_WAY_RCVD, tak sa presunie do stavu **DetAdj**, kde opäť volá funkciu *ospf_neighbor_current_event* a podľa jej výsledku sa rozhoduje, či so susedným smerovačom bude vo vzťahu ako priľahlý (adjacent) smerovač. V prípade, že dostane správu FORM_ADJ, tak sa presúva do stavu **ExStart**, kde zmení operačný stav susedného smerovača na priľahlý (posiela prázdne Database Description správy aby synchronizoval LS medzi susedmi). Vo výstupnej časti stavu opäť použije funkciu *ospf_neighbor_current_event* a v prípade, že dostane správu NEG_DONE presunie sa do stavu **Exchng**. V prípade obdržania NEIGHBOR_DOWN sa presunie do stavu **Down**, v prípade ADJ_TEARDOWN sa presunie do stavu **2-Way** a v prípade ONE_WAY_RCVD do stavu **Init**. V stave **Exchng** smerovač odošle LSA a požiadavku na LSA, v prípade, že požiadavka dorazí, proces prejde do stavu **Full** a vymenia sa aj ostatné LSA a smerovače ostanú priľahlé kým nenastane rozpad priľahlosti (prejde do stavu **2-Way**), alebo ADJ_RESET (prejde do stavu **ExStart**), alebo uplný rozpad spojenia NEIGHBOR_DOWN (prejde do stavu **Down**).

Header blok obsahuje deklarácie makier, ktoré vracia funkcia *ospf_neighbor_current_event* a funkckie:

ospf_neighbor_sv_init – Inicializácia stavových premenných pre tento proces.

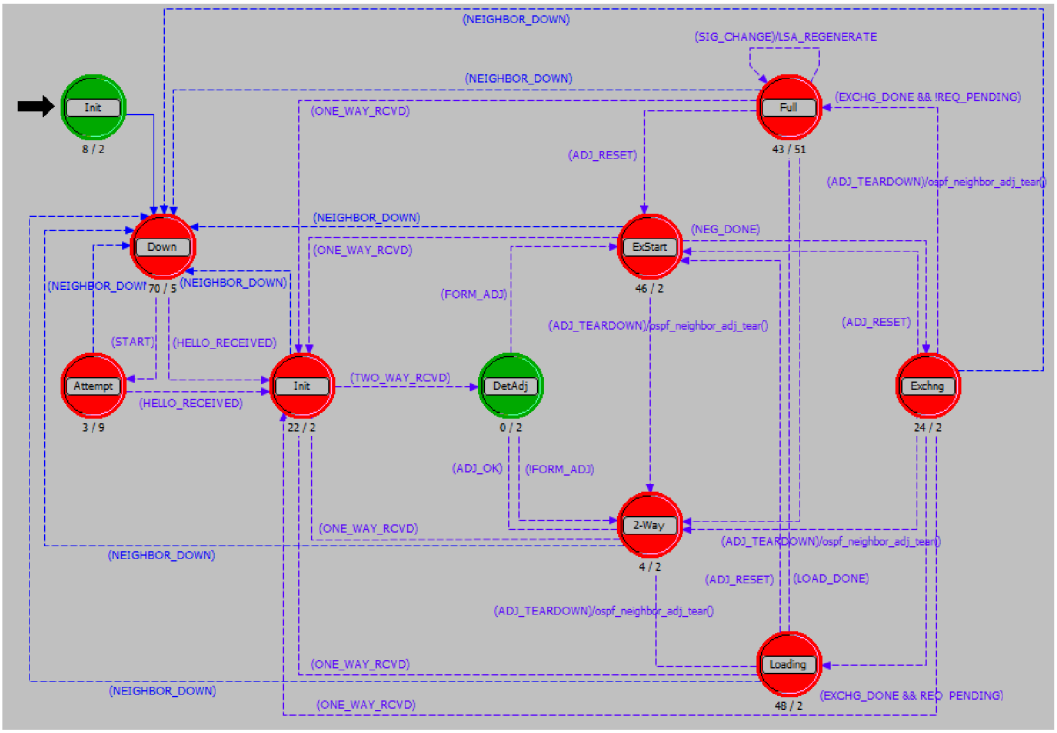
ospf_neighbor_current_event – Funkcia vráti udalosť, ktorou bol proces vyvolaný alebo naplánovaný.

ospf_neighbor_state_change – Zmení prevádzkový stav OSPF susedného procesu. Funkcia by mala byt zavolaná na začiatku každého OPNET stavu ktorý zodpovedá OSPF prevádzkovému stavu.

ospf_neighbor_inactive_timer_start – Spúšťa časovač neaktivity pre tento proces. Vypršanie tohto časovača znamená, že hello správa nebola odoslaná nedávna, čo môže značiť vypnutie susedného rozhrania. V tom prípade to vymaže susedný smerovač. Smerovač je znovu vytvorený, keď opäť dostane hello správu.

ospf_neighbor_dbase_summary_list – Zapíše obsah LSDB do zhrnutia databázy. Vykonáva sa to na začiatku link state výmeny prostredníctvom DD (Database Description) paketov.

ospf_neighbor_adj_tear – Strhnutie (čiastočne sformovanej) priľahlosti so susedným smerovačom.



Obr. 6.10: Stavový automat ospf_interface

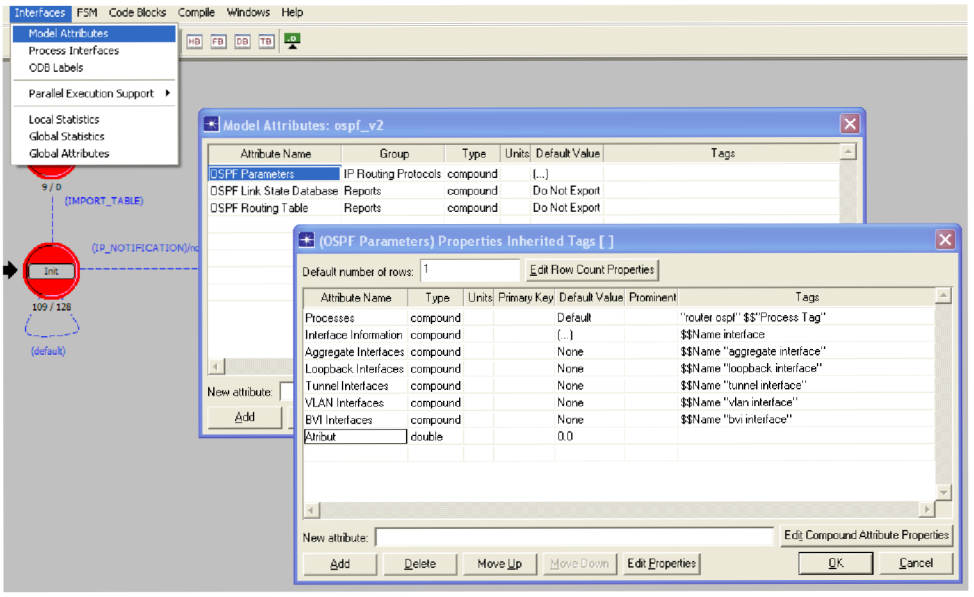
6.3 Odosielanie prenosovej rýchlosti medzi uzlami siete na OSPFv3 protokole

Podľa postupu uvedeného na začiatku kapitoly bol vytvorený model MANET siete. Cieľom tejto úlohy bolo rozšírenie Hello paketu OSPFv3 protokolu o nové pole, v ktorom sa prenáša aktuálna prenosová rýchlosť vysielacieho uzla a jej výpis na cieľovom uzle. V prípade, že by bolo potrebné prenášať v pakete atribút uzla, je text rozšírený aj o túto možnosť. Pre prenos hodnoty rýchlosti však nie je potrebné kroky týkajúce sa atribútu vykonať. V nasledujúcom texte sa nachádza postup riešenia tejto úlohy.

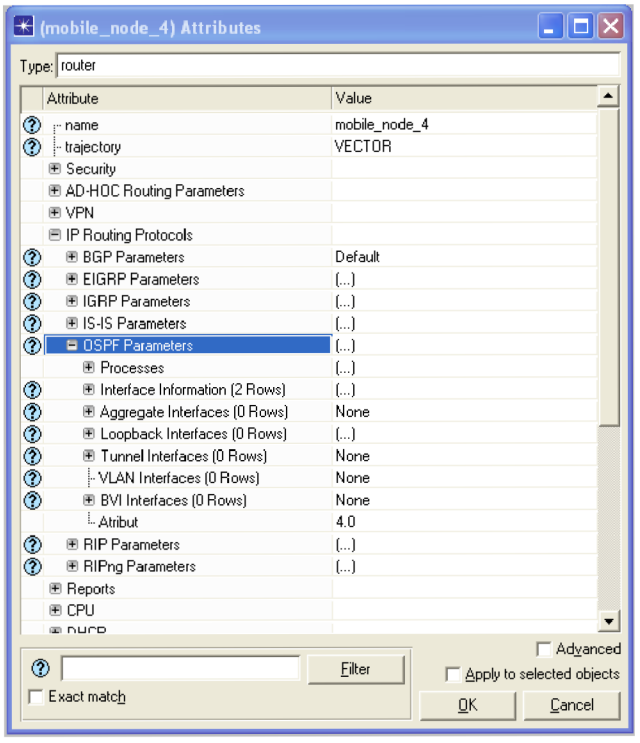
6.3.1 Vytvorenie atribútu na sieťovom uzle wlan2_router_adv

Pomocou dvojkliku na sieťový uzol sa zobrazí uzlový model daného prvku. Pokračuje sa dvojklikom na procesor **ospf**, čím sa zobrazí prvá úroveň jeho procesného modelu. Z hlavného menu sa volí ponuka **Interfaces – Model Attributes**. V otvorenom okne sa označí v stĺpci **Attribute Name OSPF Parameters** a potvrdí tlačidlom **Edit Properties**. V ďalšom okne sa vytvára nový atribút a pomocou

Edit Properties je možné ho detailne nastaviť. V tomto prípade je meno atribútu jednoducho **Atribut** a jeho typ je zvolený double s predvolenou hodnotou 0.0.



Obr. 6.11: Nastavenie atribútu



Obr. 6.12: Zobrazenie atribútu v ponuke Edit Attributes

6.3.2 Vyčítanie atribútu a jeho uloženie do stavovej premennej

Kliknutím na tlačidlo **Edit State Variables** (SV) sa zobrazí okno so stavovými premennými pre daný procesný model. Pre vloženie novej stavovej premennej sa pokračuje stlačením tlačidla **Edit ASCII**. V otvorenom okne s textom sa na záver pridá riadok:

```
1 double    \backslash premenna1;
```

6.3.3 Vyčítanie atribútu a jeho uloženie do stavovej premennej

V procesnom modeli **ospf_v2** (otvorený pomocou dvojkliku na procesor ospf v uzlovom modeli) otvoríme blok funkcií (Function Block FB) a vložíme riadok na vyčítanie hodnoty atribútu uzla a jej uloženie do stavovej premennej.

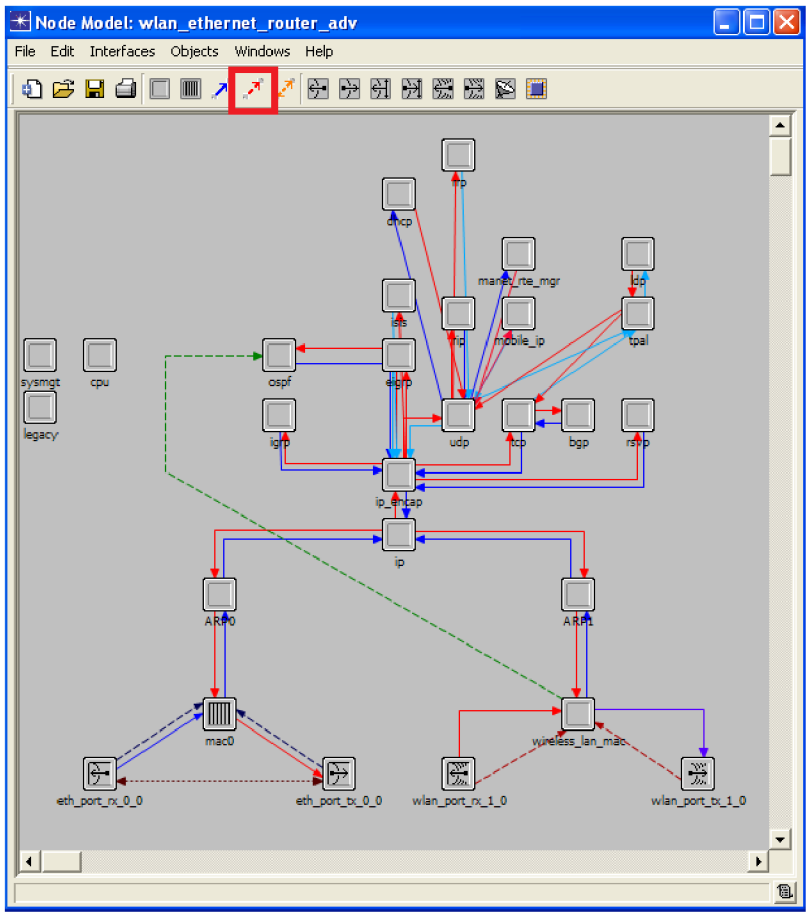
```
77 op_ima_obj_attr_get (ospf_my_params_objid, "Atribut", &premenna1);
```

Tento kód môžeme vložiť priamo do funkcie *ospf_sv_init* napr. na riadok č. 77 , pretože sa hodnota atribútu počas simulácie nebude meniť, a preto stačí jeho hodnotu vyčítať len raz. Pre kontrolu je možné pridať aj riadok s výpisom do konzoly hneď pod vyčítanie:

```
78 printf ( "Vlozeny atribut do premenna, vypis %f\n " , premenna1);
```

6.3.4 Zmeny v modeli uzlu wlan_ethernet_router

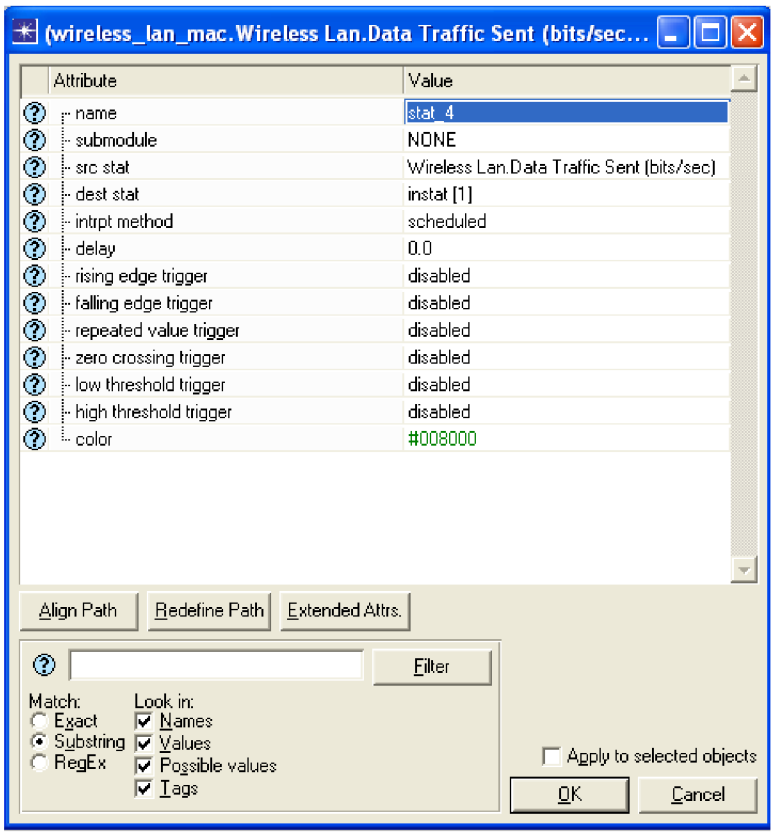
Po dvojkliku na sieťový uzol sa zobrazí jeho uzlový model. O štatistiku aktuálnej prenosovej rýchlosti v tomto modeli sa stará procesor `wireless_lan_mac`. Na predanie tejto hodnoty procesoru `ospf` sa v OM používa prvok nazvaný Statistic Wire (zvýraznená ikona na obrázku). Podľa obrázku vedieme spojenie medzi procesormi `wireless_lan_mac` a `ospf` (na obrázku zelenou farbou). Následne je ešte potrebné nastaviť atribúty tohto spojenia, preto pravým kliknutím na spojenie otvoríme ponuku, v ktorej sa volí **Edit Attributes**.



Obr. 6.13: Statistic Wire

V otvorenom okne je potrebné nastaviť nasledujúce parametre:

- **src stat – Wireless. Lan. Data.Traffic.Sent (bits/sec)** – štatistika, ktorá vyčíta aktuálnu prenosovú rýchlosť sieťového rozhrania.
- "dest stat – potrebné nastaviť na zatiaľ nepoužité číslo. V tomto prípade bola hodnota nastavená na **instat[1]**. V prípade viacerých štatistík musí mať každá tento parameter rozdielny, pretože funkcia *op_stat_local_read(x)* sa podľa indexu x rozhoduje, ktorú štatistiku má vyčítať.
- parametre prerušení je potrebné všetky prepnúť na stav **disabled**.



Obr. 6.14: Atribúty Statistic Wire

6.3.5 Vytvorenie súboru na zápis prenosovej rýchlosti

Pre každý sieťový uzol bolo potrebné vytvoriť samostatný súbor pre zápis prenosovej rýchlosti a názvu uzlu. Dané súbory bolo najjednoduchšie vytvoriť priamo pri štarte simulácie, preto bol kód na tvorbu súboru zapísaný na začiatok funkcie *ospf_sv_init*. K danej funkcii sa dostaneme dvojklikom na procesor `ospf` a otvorením FB, v ktorom funkcia začína na riadku č. 51. Medzi premenné lokálne deklarované vo funkcii pridáme pole znakov meno dĺžky 32 a pointer na súbor *f:

```
51 char   meno[32];
52 FILE   *f;
```

Do tela funkcie na riadok č. 78(OM v16 – 161) bolo potrebné pridať nasledujúci kód:

```
78 op_ima_obj_attr_get (op_topo_parent (op_id_self ()), "name", &meno)
      ;
79 strcat ( meno, ".txt");
80 printf("vytvaram subor %s\n",meno);
```

```
81
82  if ((f = fopen ( meno ,"w")) == NULL )
83
84    {
85    printf (" Subor sa nepodarilo otvorit \n");
86     }
87    fprintf (f,
88    "|-----------------|---------------|--------------------| \n"
89    "|       Cas       |      Uzol     | Prenosova rychlost | \n"
90    "|-----------------|---------------|--------------------| \n");
91
92  if ( fclose (f) == EOF )
93    {
94    printf (" Subor sa nepodarilo zatvorit \n");
95     }
```

Týmto krokom je ukončená úprava funkčného bloku ospf_v2.

6.3.6 Pridanie poľa do Hello paketu

V súborovej štruktúre OM, presnejšie v zložke **OPNET\1x.x.x\models\std\include**, sa nachádza súbor **ospf_pkt_sup.h**. Vytvoríme záložnú kópiu tohto súboru a súbor otvoríme a pridáme riadok:

```
51 double  rychlost;
```

do štruktúry Ospft_hello_pkt_Fields (príkaz je umiestnený na 51. riadku súboru).Ďalej je potrebné upraviť funkcie v súbore **ospf_pkt_sup.ex.c (OPNET\1x.x.x\models\std\ospf\ospf_pkt_sup.ex.c)**:
ospf_hello_pkt_fdstruct_create a *ospf_hello_pkt_fdstruct_print*.
Teda do *ospf_hello_pkt_fdstruct_create* doplniť nasledujúci kód na riadok 119:

```
119 pk_fdstruct_ptr->rychlost   = 0.0;
```

a vo funkcii ospf_hello_pkt_fdstruct_print pred príkaz

```
1 FOUT;
```

pridať 2 riadky na vypisovanie hodnoty v pakete v OPNET Debuggeri:

```
1 sprintf (temp_str, "Rychlost: %f ", pk_fd_ptr->rychlost);
2 PKPRINT_STRING_INSERT (alloc_str, temp_str, list_ptr);
```

6.3.7 Vyčítanie prenosovej rýchlosti a jej vkladanie do paketu

Nasledujúce úpravy bolo potrebné vykonať vo funkčnom bloku dcérskeho procesu ospf_interface_v2. Za ideálne miesto na vyčítanie prenosovej rýchlosti bola zvolená funkcia *ospf_interface_hello_timer_set*, pretože práve v nej sa volá funkcia *ospf_message_hello_create*, slúžiaca na tvorbu Hello paketu. Vo funkcii *ospf_interface_hello_timer_set* bolo potrebné lokálne deklarovať premennú:

```
1 double    rychlost;
```

do ktorej následne pred volaním funkcie *ospf_message_hello_create* na riadku 287 vloží hodnota prenosovej rýchlosti pomocou nasledujúceho kódu:

```
284 rychlost=op_stat_local_read(1);
285 printf("rychlost rozhrania je %f\n", rychlost);
```

Ďalej bolo potrebné obsah tejto premennej predať funkcii ospf_message_hello_create, preto sa riadok 287:

```
287 hello_message_ptr = ospf_message_hello_create (
      ospf_interface_self_ptr);
```

rozšíri na:

```
287 hello_message_ptr = ospf_message_hello_create (
      ospf_interface_self_ptr, rychlost);
```

Rovnaké úpravy ako vo funkcii *ospf_interface_hello_timer_set* , bolo potrebné vykonať aj vo funkcii *ospf_interface_mdr_hello_send*, pretože tiež volá funkciu *ospf_message_hello_create*. Týmto krokom sa však zmenila deklarácia funkcie a preto ju bolo potrebné pozmeniť. Deklarácia tejto funkcie sa nachádza v súbore **ospf_msg.h** a jej definícia v súbore **ospf_msg_v2.ex.c**. V súbore **ospf_msg.h** funkciu nachádzajúcu sa na 22. riadku rozšírime preberaný parameter typu double.

```
22 ospf_message_hello_create (OspfT_Interface *interface_ptr, double);
```

V súbore **ospf_msg_v2.ex.c** bolo potrebné upravit definícou funkcie aby zodpovedala deklarácii. Preto do definície bolo dopísané double rychlost.

```
22 ospf_message_hello_create (OspfT_Interface *interface_ptr, double
      rychlost)
```

Poslednou úpravou súboru **ospf_msg_v2.ex.c** je vloženie hodnoty parametru do položky štrukúry na riadku 107(v16 – 117).

```
107 ospf_pk_fd_ptr->rychlost  = rychlost;
```

Týmto krokom sa bude v poli paketu prenášať hodnota aktuálnej prenosovej rýchlosti, ktorá sa dá zobraziť odchytením paketu v OM Debuggeri.

6.3.8 Získanie atribútu v dcérskom procese ospf_interface_v2

V prípade, že je potrebné vyčítať atribút je postup nasledovný:
Vo funkcii *ospf_interface_mdr_hello_send* je potrebné lokálne deklarovať premenné:

```
1 int        myid;
2 int        parentID1;
3 char       name [128];
4 double *   parametre;
```

a vyčítenie atribútu sa potom vykoná pomocou nasledovného sledu príkazov, ktoré je potrebné umiestniť pred volanie funkcie *ospf_message_hello_create*:

```
285
286 /*získanie ID vlastného procesu*/
287 myid = op_id_self ();
288 /*získanie ID otcovského procesu*/
289 parentID1 = op_topo_parent (myid);
290 /*získanie mena otcovského uzlu */
291 op_ima_obj_attr_get (parentID1, "name", &name);
292 /* výpis mena otcovského uzlu*/
293 printf ("Vlozeny atribut name, vypis: %s \n ", name);
294 /*získanie ukazateľa na SV premenna1 a jej uloženie do premennej
      parametre*/
295 parametre = (double *) op_ima_obj_svar_get (myid, "premenna1");
296 /*výpis premennej na kontrolu správnosti*/
297 printf ("Vlozeny atribut do premenna1, vypis %f \n ", * parametre);
```

a rozšíriť parametre, ktoré preberá nasledujúca funkcia *ospf_message_hello_create* o premennú *parametre, teda výsledný riadok bude vyzerať nasledovne:

```
297 hello_message_ptr = ospf_message_hello_create (
      ospf_interface_self_ptr, * parametre);
```

Rovnakú úpravu je potrebné vykonať aj pred druhom volaní funkcie vo funkčnom bloku. Ukážka takto upraveného paketu je na nasledujúcom obrázku.

```
Packet content


_____________________________ (ODB 14.0.A: Event) _____________________________

    * Time   :  17.30577783685 sec, [17s . 305ms 777us 836ns 850ps]
    * Event  :  execution ID (5948), schedule ID (#6100), type (stream intrpt)
    * Source :  execution ID (5947), top.Office Network.mobile_node_4.ip [Objid=6318] (processor)
    * Data   :  instrm (0), packet ID (2045)
    > Module :  top.Office Network.mobile_node_4.ip_encap [Objid=6692] (processor)
              ID              : 2044
              tree ID         : 294
              address         : 0x045BCB08
              format          : ospf_hello_v3
              creation module : top.Office Network.mobile_node_3.ospf [Objid=4688]
              creation time   : 17.305490715336 sec. [17s . 305ms 490us 715ns 336ps]
              stamp module    : top.Office Network.mobile_node_3.ospf [Objid=4688]
              stamp time      : 17.305490715336 sec. [17s . 305ms 490us 715ns 336ps]
              bulk size       : 0 bits
              total size      : 352 bits
              owner           : top.Office Network.mobile_node_4.ip_encap_to_ospf [Objid=7495]
              ICI ID          : NONE
              ID trace        : off
              tree ID trace   : off
              encap flags     : NONE
_____________________________________________________________________________
Packet Fields
              Index Name                Type        Value           Size
              ----- ------------------- ----------- --------------- ----
                  0 type                integer     1               8
                  1 fields              structure   0x02DDE518      160

 fields structure from OSPF hello packet

 version: 3, instance: 1

 Hello Interval: 10.000000, Router Dead Interval: 40.000000

 priority: 1, options: 514

 Router ID: 192.0.0.8, Area ID: 0.0.0.0, Network Mask: 0.0.0.1

 DR: 0.0.0.0, Backup DR: 0.0.0.0


 Rychlost: 3.000000

                  2 header info         information unspecified     120
                  3 field_3             structure   0x045148D8      64
_____________________________________________________________________________
Transmission Data Attributes (TDA)
              No associated TDAs.
```

Obr. 6.15: Upravený Hello paket

6.3.9 Výpis prenosovej rýchlosti do súboru

Spracovanie všetkých príchodzích paketov sa vykonáva v dcérskom procese **ospf_process** vo funkcii *ospf_root_message_dispatch*, ktorá sa stará o rozlišovanie typu paketu a podľa toho volá funkcie na spracovanie jednotlivých paketov. Pre túto úlohu bolo dôležité spracovanie Hello správ, preto vo funkcii nájdeme obsluhu práve týchto správ. V switchi sa v prípade Hello správy volá funkcia *ospf_hello_message_rcvd*, ktorá bola následne rozšírená o blok príkazov na výpis do súboru.

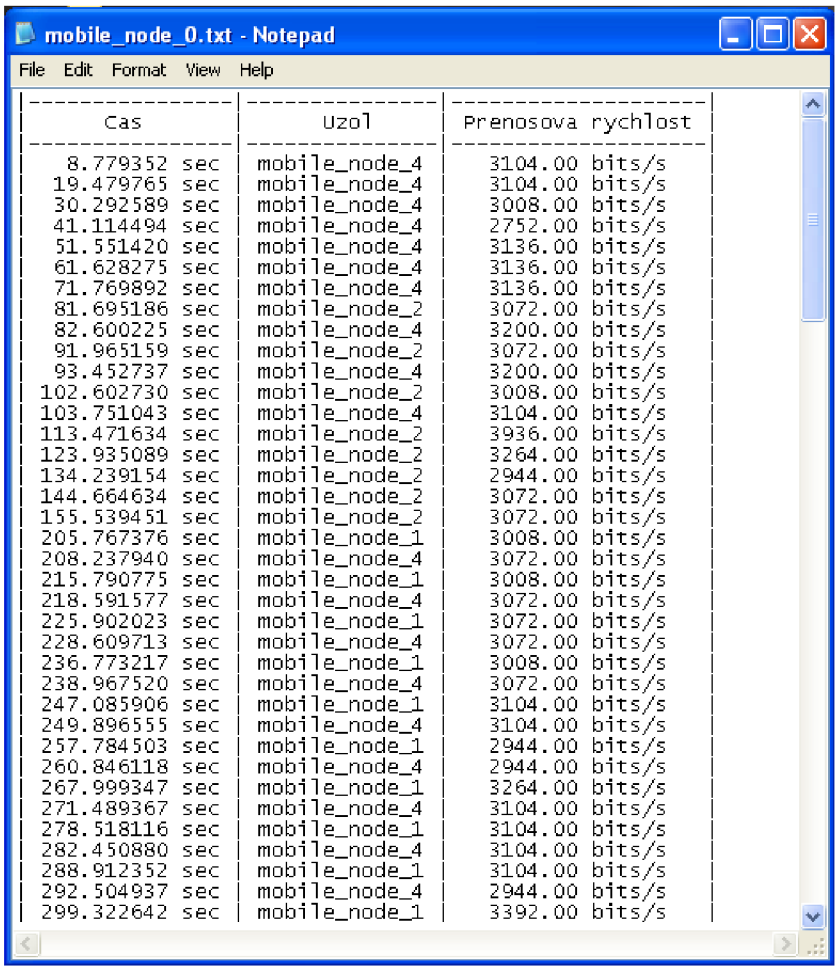
Deklarácia premených v tejto funkcii bola rozšírená o:

```
1 char     meno[30];
2 char     zdroj_meno[32];
3 Objid    zdroj_modul;
4 Objid    zdroj_uzol;
5 double     cas;
6 FILE     *f;
```

A telo funkcie o blok príkazov:

```
1252
1253   cas=op_sim_time();
1254   zdroj_modul = op_pk_creation_mod_get (message_ptr);
1255   zdroj_uzol =  op_topo_parent (zdroj_modul);
1256   op_ima_obj_attr_get (zdroj_uzol, "name", &zdroj_meno);
1257   op_ima_obj_attr_get (op_topo_parent (op_id_self ()), "name", meno
          );
1258   printf("ja %s som dostal som hello od %s v case %f. rychlost je %
          f \n",meno,zdroj_meno,cas,pk_fd_ptr->rychlost);
1259
1260   strcat (meno, ".txt");
1261   printf("otvaram subor %s\n", meno);
1262
1263
1264   if ((f = fopen ( meno ,"a")) == NULL )
1265     {
1266   printf (" Subor sa nepodarilo otvorit \n");
1267     }
1268       fprintf (f,  "| %10.6f sec | %s |   %.2f bits/s   | \n", cas,
              zdroj_meno, pk_fd_ptr->rychlost );
1269   if ( fclose (f) == EOF )
1270     {
1271     printf (" Subor sa nepodarilo zatvorit \n");
1272     }
```

Tie bolo potrebné napísať na riadok 1252 (OM v16 – 1270) za funkciu *op_pk_nfd_access*. V danom bloku sa ako prvý údaj získa aktuálny čas simulácie, a Objid modulu v ktorom sa došlá správa vytvorila. Pomocou funkcie *op_topo_parent* sa získa Objid uzla, ktorý správu vytvoril. Následne sa získa meno uzla a meno uzla, ktorý správu dostal. Potom už nasleduje iba obsluha zápisu daných údajov do súboru pomenovanom podľa uzla, ktorý správu dostal. Na nasledujúcom obrázku sa nachádza ukážka súboru s textovým výstupom programu.



Obr. 6.16: Ukážka súboru s textovým výstupom programu

ZÁVER

V bakalárskej práci sú zahrnuté teoretické poznatky, týkajúce sa základných znalostí ohľadom funkcie mobilných sietí so zameraním na MANET siete a smerovacie protokoly v nich použité. Práca je rozdelená do 6 kapitol. Prvá kapitola popisuje rozdelenie v súčasnosti používaných mobilných sietí a ich topológii. Druhá kapitola je zameraná na smerovanie v dátových sieťach a prezentuje rôzne prístupy k smerovaniu, či už v sieťach káblových alebo bezdrôtových. Tretia kapitola je venovaná IPv6 protokolu, keďže práve na ňom pracuje protokol OSPFv3, ktorému je venovaná najrozsiahlejšia kapitola v teoretickej časti práce. Obsahuje rozdelenie smerovacích oblastí a druhy smerovačov v nich. Taktiež sú v nej uvedené druhy paketov, ktoré sa v tomto protokole používajú. V piatej časti sa nachádza krátky popis prostredia OPNET modeler a jeho editorov.

Posledná šiesta kapitola je zameraná na popis stavby modelu siete pracujúcej na protokole OSPFv3 v OPNET modeleri verzie 16. Samotná kapitola sa dá rozdeliť na 3 podkalpitoly. Prvá z podkapitol obsahuje podrobný popis stavby modelu vrátane obrázkov znázorňujúcich polohu jednotlivých nastavení. Druhá podkapitola sa zaoberá procesným modelom s popisom stavov a funkcií procesoru ospf a jeho dcérskych procesov. Posledná časť je venovaná postupu rozšírenia OSPFv3 Hello paketu a poskytuje riešenie prípadov, keď je nutné v pakete prenášať určitú hodnotu, či už sa jedná o atribút nastavený na sieťovom uzle alebo o štatistiku. Tieto údaje by sa v budúcnosti dali použiť na zabezpečenie kvality služieb QoS v sieti MANET s protokolom OSPFv3.

LITERATURA

[1] Bouška, P. *Cisco Routing 3 - OSPF - Open Shortest Path First.* [cit. 2010-12-13]. Dostupné na: <http://www.samuraj-cz.com/clanek/cisco-routing-3-ospf-open-shortest-path-first>.

[2] Doyle, J. a Carroll, J. *Routing TCP-IP Volume 1.* [b.m.]: Cisco Press, 2005. 936 s. ISBN 1-58705-202-4.

[3] Hong, X., Xu, K. a Gerla, M. *Scalable routing protocols for mobile ad hoc networks.* 2002. Dostupné na: <http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.76.9545&rep=rep1&type=pdf>.

[4] Kolektiv autorov. *OSPF Version 2.* 1998. Dostupné na: <http://www.ietf.org/rfc/rfc2328>.

[5] Kolektiv autorov. *OSPF for IPv6.* 2008. Dostupné na: <http://tools.ietf.org/html/rfc5340>.

[6] Molnár, K. *Hardware počítačových sítí : Skriptum.* [b.m.]: VUT v Brně. 110 s.

[7] Molnár, K., Zeman, O. a Skořepa, M. *Moderní síťové technologie : Laboratorní cvičení.* [b.m.]: VUT v Brně, 2008. 101 s. Dostupné na: <http://www.utko.feec.vutbr.cz/~molnar/mmos/MMOS_lab.pdf>.

[8] Novotný, V. *Mobilní směrovací protokoly s podporou IPv6 (MANET).* [b.m.]: VUT v Brně, 2007. 77 s. Dostupné na: <http://docs.google.com/viewer?a=v&pid=wave&srcid=8QI35XxE1&chrome=true&pli=1>.

[9] OPNET Technologies, Inc.. *Understanding MANET Model Internals and Interfaces, OPNET tutorial 1941.* 2007.

[10] OPNET Technologies, Inc.. *OPNET Modeler Product Documentation Release 16.0.* 2010.

[11] Raab, S. a Chandra, M. *Mobile IP Technology and Applications.* [b.m.]: Cisco Press, 2005. 312 s. ISBN 1-58705-375-6.

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

ABR Area Border Router

AODV Ad hoc On Demand Distance Vector Routing

AP Access Point

AS Autonomous system

ASBR Autonomous System Boundary Router

BDR Backup Designated Router

BGP Border Gateway Protocol

BR Backbone router

CGSR Cluster-Head Gateway Switch Routing Protocol

CIDR Classless Inter-Domain Routing

DD Database Description

DR Designated Router

DREAM Distance Routing Effect Algorithm for Mobility

DSR Dynamic Source Routing Protocol

DV Distance Vector

FB Function Block

FSLS Fuzzy Sighted Link State Algorithm

FSR Fisheye State Routing Protocol

GPSR Greedy Parameter Stateless Routing

GSM Global System for Mobile Communications

HB Header Block

HSR Hierarchical State Routing

HTML Hypertext markup language

IETF Internet Engineering Task Force

IOS Internetwork Operating System

IP Internet Protocol

IPv6 Internet Protocol, Version 6

IR Internal router

LANMAR Landmark Routing Protocol

LAR Location Aided Routing

LS Link-State

LSA Link State Advertisement

LSDB Link State Database

MANET Mobile Ad hoc Network

MOSPF Multicast Open Shortest Path First

NSSA Not-So-Stubby Area

OLSR Optimized Link State Routing Protocol

OM OPNET Modeler

OSPF Open Shortest Path First

OSPFv3 Open Shortest Path First, Version 3

QoS Quality of Service

RIP Routing Information Protocol

RIP2 Routing Information Protocol, Version 2

RPF Reverse Path Forwarding

SPF Shortest Path First

SV State Variable

TBRPF Topology Broadcast based Reverse-Path Forwarding

TCP Transmission Control Protocol

TCP/IP Transmission Control Protocol/Internet Protocol

TORA Temporally-Ordered Routing Algorithm

TTL Time to Live

UDP User Datagram Protocol

VLSM Variable-Length Subnet Mask

XML Extensible Markup Language

ZRP Zone Routing Protocol

A PRÍLOHA

A.1 Obsah CD

- Bakalárska práca vo formáte pdf
- Zdrojové súbory pre LaTeX
- Projekt siete s upravenou Hello správou pre OM v16
- Projekt siete s upravenou Hello správou pre OM v14
- Projekty sietí s rôznym počtom uzlov a rýchlosťou pohybu
- Upravené zdrojové kódy v16
- Upravené zdrojové kódy v14