VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

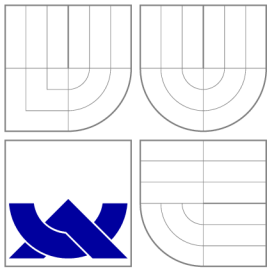
MODELOVÁNÍ LINK-STATE SMĚROVACÍHO PROTOKOLU OSPFV3

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

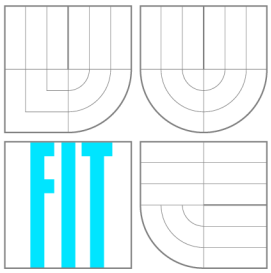
AUTOR PRÁCE
AUTHOR

Bc. JAKUB MRÁZEK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ LINK-STATE SMĚROVACÍHO PROTOKOLU OSPFV3

MODELLING OF OSPFV3 LINK-STATE ROUTING PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAKUB MRÁZEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2014

Abstrakt

OMNeT++ je diskrétní modulární simulátor, často využívaný pro simulaci počítačových sítí. Univerzální simulátor lze doplnit různými rozšiřujícími balíky např. INET Framework, který slouží pro simulaci zařízení a protokolů síťového zásobníku TCP/IP. V této práci, zkoumáme možnosti balíku INET v oblasti směrování, tedy dynamické směrovací protokoly. Konkrétně se práce zaměřuje na protokol OSPF pro IPv6 sítě (OSPFv3). Protokol je v této práci představen a jsou zde vysvětleny principy jeho fungování. V práci je navrhnut modul OSPFv3 pro simulování protokolu a je z části implementován.

Abstract

OMNeT++ is a discrete event, modular simulator often used for simulation of computer networks. The universal simulator can be extended by various packages, such as INET framework, which is used for simulation of devices and TCP/IP protocols. This thesis examines facilities of INET package in the routing that is dynamic routing protocols. Concretely, it aims at OSPF protocol for IPv6 networks (OSPFv3). The protocol is presented and its principles are interpreted. The work is designed to simulate the module OSPFv3 protocol and module is partially implemented.

Klíčová slova

Link-state směrovací protokol, OSPFv3, síťová simulace, diskrétní simulace, OMNeT++, INET framework

Keywords

Link-state routing protocol, OSPFv3, network simulation, discrete simulation, OMNeT++, INET framework

Citace

Jakub Mrázek: Modelování link-state směrovacího
protokolu OSPFv3, diplomová práce, Brno, FIT VUT v Brně, 2014

Modelování link-state směrovacího protokolu OSPFv3

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Veselého. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Jakub Mrázek
28. května 2014

Poděkování

Tímto bych chtěl poděkovat úplně všem, především vedoucímu práce Ing. Vladimíru Veselému za po poskytování neocenitelných rad. Dále bych chtěl poděkovat přítelkyni a rodičům a našemu pejskovi za udržování mé duševní a fyzické stránky v rozumném rozpoložení. Všem přátelům, které jsem v průběhu studia potkal a zpříjemnili mi studium. A také zahraničním studentům z Erasmu, kteří pro mě byli rodinou po dobu zahraničního studia.
V neposlední řadě bych chtěl poděkovat této skvělé buchtě, která mi úspěšně doplňovala energii, a i navazovala vztahy s přáteli v zahraničí.

Suroviny:

- Těsto: 40 dkg hladké mouky, 15dkg práš.cukru, 2 celá vejce, 0.5 prášku do pečiva, 1 máslo
- Náplň: 2 lžíce rumu, 2 celá vejce, 2 tvarohy měkké(0.5 kg), 15dkg práškového cukru, 1 vanilkový cukr, 1 vanilkový puding, 0.5 l mléka

Těsto se rozválí na plech a dělají se vyšší okraje. Náplň je tekutá, po upečení ztuhne. Doporučuji přidat na těsto kusy ovoce nebo čtverečky čokolády a pak teprve zalít náplní.

Obsah

Kapitola 1

Úvod

Složitost sítí a používaných technologií stále roste. Projekt OMNeT++ poskytuje vývojové a simulační prostředí pro diskrétní simulace sítí. Pro modelování lze použít některý z dostupných rozšíření, například framework INET, který se zaměřuje na TCP, UDP a IP sítě. Projekt ANSA na FIT VUT v Brně se zabývá rozšiřováním funkcionality INET se snahou obsáhnout komplexní modely síťových zařízení, které by bylo možné využít pro formální analýzu sítí. Tato práce se zabývá modelováním protokolu OSPF pro IPv6 v prostředí OMNeT++.

S příchodem Internetu se počítačové sítě rozrostly do velkých rozměrů. Nebylo v možnostech správců sítí udržovat tyto sítě a cesty mezi nimi ve statické podobě. Začaly se tedy rozvíjet dynamické směrovací protokoly. Nejdříve vznikly protokoly, které nejlepší cestu od směrovače do dalších sítí určovaly počtem směrovačů v cestě (protokol RIP). Tento jednoduchý způsob byl zdokonalen přidáním granulárnější metriky (protokol EIGRP). Ty jsou společně označovány, protože určují cestu podle vzdálenosti, jako *distance-vector*. Mimo tuto skupinu protokolů, vznikla skupina protokolů nazývaná *link-state*, které při výpočtu cesty znají topologie celé sítě (stavy linek). Do této skupiny patří OSPF a IS-IS. Ty jsou daleko složitější, jak na implementaci, tak na použití. Nabízí za to lepší funkcionalitu: větší flexibilitu pro jiné protokoly, rychlejší reakci na změny v síti a lépe vybrané cesty v síti, které se použijí pro směrování.

Kapitola 2

IPv6 Protokol

Tato kapitola vysvětluje protokol IP (*Internet Protocol*) verze 6 (IPv6). Směrovací protokol OSPFv3, kterému se věnuje tato práce, podporuje IPv6, pro tuto práci je nejdůležitější adresování v IPv6.

Starší protokol IPv4 je nejrozšířenější protokol IP. Nad tímto protokolem byl vystavěn Internet, byl popsán v RFC 791 (*Request For Comments*) v roce 1981 [14]. Tento protokol přestal v současnosti vyhovovat požadavkům, především omezeným počtem připojených uzlů (počítačů, směrovačů, atd.). Jeho adresy jsou dlouhé 32 bitů, což umožňuje teoreticky připojit 4 294 967 295 uzlů, ovšem některé adresy jsou vyhrazeny pro jiné účelové použití (multicast, lokální adresy atd.). Tento prostor ještě pomohl zvětšit NAT (*Network Address Translation*), který umožňuje překládat lokální adresy na jednu nebo více veřejných adres. To ovšem omezuje použití některých vyšších protokolů.

Tyto problémy rozsahu a překládání adres má odstranit protokol IPv6, poprvé specifikován v RFC 1883 [10] a dalších RFC z roku 1995, který má adresy dlouhé 128 bitů a počet adres je v současnosti považován za dostatečně velký pro připojení všech uzlů s velkou rezervou do budoucnosti. Nepředpokládá se vyčerpání.

2.1 Zápis adresy IPv6

Jak již bylo uvedeno výše nejzásadnější změnou přinášející protokol IPv6 je rozšíření adresního prostoru na 128 bitů oproti 32 bitům protokolu IPv4. Adresování je specifikováno v současnosti platném RFC 4291 [11] z roku 2006.

V IPv4 je adresa zapsáná v desítkové soustavě tak, že každý bajt (8 bitů) je oddělen tečkou např.: 192.168.0.12. V IPv6 je adresa zapsaná v šestnáckové soustavě tak, že dvojice bajtů (16 bitů) jsou odděleny dvoutečkou. Dvojice bajtů je zapsaná čtyřmi symboly šestnáckové soustavy (0-f). Protože adresy i přes zápis v šestnáckové soustavě jsou velmi dlouhé je umožněno zkracování adres, tak že se nemusí psát nuly na začátku dvojice bajtů např. prefix:0001:sufix zapíšeme jako :1: a jedenkrát lze zkrátit několik nulových po sobě následujících dvojicí bajtů tak, že se nezapíše nic např.: prefix:0000:0000:sufix na prefix::sufix. Názorný příklad je uvedený v tabulce 2.1, výsledná adresa je např.: 20af:0:0:201:a4::1. Všechny tyto varianty z tabulky jsou platné a označují stejnou adresu, i když nuly jsou zkráceny na různých místech. Platí to, protože adresu zle doplnit nulami na její délku. Nesmíme ovšem adresu zkrátit na dvou a více místech, protože by bylo nejasné kolik nul do zkrácení doplnit a kde se přesně čísla mezi nulami nachází.

V IPv4 se pro zápis masky používala stejná notace jako adresy a nebo se využíval kratší

20af	:	0000	:	0000	:	0201	:	00a4	:	0000	:	0000	:	0001
20af	:	0	:	0	:	201	:	a4	:	0	:	0	:	1
20af	:				:	201	:	a4	:	0	:	0	:	1
20af	:	0	:	0	:	201	:	a4	:				:	1

Tabulka 2.1: Zkracování adresy IPv6

zápis pouze počtem platných bitů od začátku adresy uvedený za lomítkem např. maska 255.255.128.0 může být uvedena jako adresa/17.

V IPv6 se již využívá pouze zápis za lomítkem, protože označuje délku adresy od začátku, je označována jako délka prefixu (*prefix length*) a adrese sítě se říká prefix, má platné pouze bity určené délkou prefixu (maskou). Příklad adresy s délkou prefixu: 20af:0:0:201:a4::1/64 a prefixu s délkou prefixu (adresy sítě): 20af:0:0:201::/64.

2.2 Typy adresy IPv6

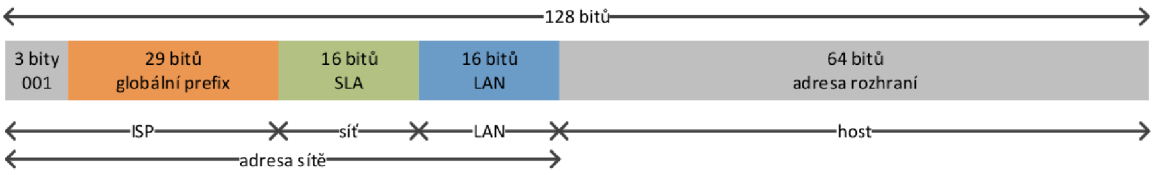
Jak již bylo uvedeno v úvodu této kapitoly IPv6 obsahuje tolik adres, že se nepředpokládá vyčerpání. Adresy jsou rozděleny dle prefixů pro různé využití, tyto specifické prefixy jsou uvedeny v tabulce. Dále adresy můžeme zjednodušeně rozdělit podle platnosti a podle způsobu doručení ([11] sekce 2.5-2.7), jak je uvedeno v tabulce 2.2.

prefix/délka prefixu	využití prefixu
::/128	nespecifikovaná
::1/128	loopback
2000::1/3	globální
FC00::1/7	lokální
FE80::/10	linková
FF00::/8	multicast

Tabulka 2.2: Prefixy a využití

Dělení adres dle platností regionů:

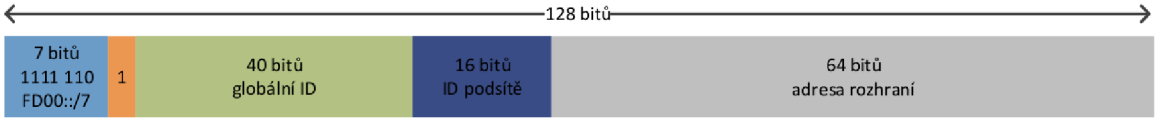
- ***Veřejné adresy*** (*Global Address*)
  Tato adresa je platná v rámci celého Internetu.
  Veřejná adresa se skládá z bitů v pořadí takto:



Obrázek 2.1: Rozdělení IPv6 veřejné adresy (obrázek přejat z [20])

- 3 bity - První tři byty jsou nastaveny na 001, zbylé varianty jsou ponechány do budoucnosti s tím, že pro tyto adresy může být zvolen jiný způsob přidělování a využívání.
- 29 bitů - Je globální prefix, který přiděluje IANA[1] jednotlivým poskytovatelům (*service providers*).
- 16 bitů - SLA (Site Level Aggregator) přiřazený koncovému uživateli poskytovatelem.
- 16 bitů - LAN (Local Area Network) části, která reprezentuje podsíť v rámci sítě koncového uživatele.
- 64 bitů - Je použito pro určení adresy rozhraní.

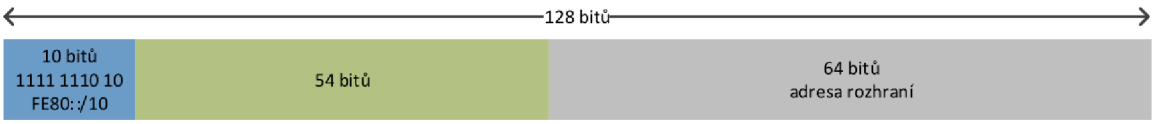
- ***Lokalní adresy*** (*Unique Local Address*)
  Adresa s prefixem FD00::/7 se používá na lokální síti a není směřována v Internetu, obdoba privátní adresy v IPv4.
  Veřejná adresa se skládá z bitů v pořadí tak:



Obrázek 2.2: Rozdělení IPv6 lokální adresy (obrázek přejat z [20])

- 7 bitů - Prvních sedm bytů je nastaveno na 1111 110 (FD00::/7).
- 1 bit - Nastaven na 1 lokálně přiřazeno a 0 - nespecifikováno, ponecháno pro budoucí rozšíření.
- 40 bitů - Globální ID nastavuje správce lokální sítě.
- 16 bitů - ID podsítě, nastavuje také správce lokální sítě, slouží pro vytvoření hierarchického systému adres.
- 64 bitů - Je použito pro určení adresy rozhraní.

- ***Linkové adresy*** (*Link Local Address*)
  Linková adresa s prefixem FE80::/10, která se automaticky přidělí rozhraní bezstavovou autokonfigurací adres popsanou v RFC 4862 [19]. Tato adresa se nepřenáší mimo linku.



Obrázek 2.3: Rozdělení IPv6 linkové adresy (obrázek přejat z [20])

[1]IANA - Internet Assigned Numbers Authority, http://www.iana.org

  - 10 bitů - Prvních deset bytů je nastaveno na 1111 1110 10 (FE80::/10).
  - 40 bitů - Nastaveno na nulu nebo libovolnou hodnotu.
  - 64 bitů - Je použito pro určení adresy rozhraní, automaticky generovaná.

- ***Multicastové adresy*** (*Multicast Address*)
  Linková adresa s prefixem FF00::/8, Adresy, které má skupina uzlů v síti a pakety zaslané na tuto adresu jsou doručeny všem uzlům s touto adresou adresou.
  - 4 bity - Životnost nastavena na 0 - trvalá adresa (dobře známá) a 1 - dočasná adresa.
  - 4 bity - Rozsah platnosti, příklady rozsahu uvedeny níže:
    * 1 - Uzel.
    * 2 - Linka.
    * 4 - Správcem určená lokální síť.
    * 5 - Lokální síť.
    * 8 - Organizace.
    * E - Globální - celý Internet.
  - 112 bitů - Je použito pro určení adresy skupiny.

  Některé definované multicastové adresy podle IANA [2], jsou v uvedeny v tabulce 2.3.

FF01::1	Uzel
FF02::1	Uzly na lince
FF01::2	Směrovač
FF02::2	Směrovače na lince
FF05::1	Směrovače v lokální síti
FF02::5	OSPF Směrovače
FF02::6	OSPF DR Směrovače
FF02::9	RIP Směrovače
FF02::A	EIGRP Směrovače

Tabulka 2.3: Definované multicastové adresy

Kapitola 3

Principy OSPF

V této kapitole budou vysvětleny principy používající OSPF verze 2 pro IPv4 [1]. Většina těchto technik je stejná v OSPF verze 3 pro IPv6. Změny, které jsou v OSPFv3, vysvětluje další kapitola 4.

3.1 OSPFv2

Open Shortest Path First (OSPF) protokol je definovaný v RFC 2328 [13]. Slouží k distribuci směrovacích informací uvnitř jednoho autonomního systému (Interior Gateway Protocol, IGP). OSPF je směrovací protokol patřící do rodiny protokolů TCP/IP. Na rozdíl od staršího směrovacího protokolu RIP, který využívá pro určení počet směrovačů v cestě (*distance-vector routing protocols*) a cestu určuje algoritmem Bellman-Ford, OSPF využívá znalosti stavu linek v celé síti (*link-state routing protocols*) a cestu počítá Dijkstrovým algoritmem. Stav linek jsou informace z rozhraní směrovačů o ceně linky, IP adrese. Tyto informace jsou uloženy v link-state databázi. OSPF zavedlo nové vlastnosti jako ověřování směrovacích informací, proměnou délku podsítě (Varible Length Subnet Masks, VLSM), sumarizaci cesta, rozdělení sítě do hierarchických částí, odesílání častí směrovacích informací a další.

3.2 SPF algoritmus

OSPF používá Shortest Path First algoritmus, kterým je vypočítána nejkratší cesta ke všem známým cílům. Tento algoritmus pro výpočet je Dijkstrův algoritmus.

Přehled SPF algoritmu

- Po inicializaci nebo změně směrovacích informací, směrovač generuje link-state inzerci (*advertisement*) LSA. Inzerce obsahuje všechny linky připojené na tomto směrovači.
- Všechny směrovače odesílají link-state aktualizaci (*update*) LSU. Po obdržení aktualizace si směrovač uloží informace do své link-state databáze LSDB, a poté šíří aktualizaci dalším směrovačům v síti.
- Poté, co je upravena link-state databáze, směrovač počítá cestu do všech sítí. Pro výpočet se používá Dijkstrův algoritmus pro určení nejkratší cesty ve stromu. IP adresa sítí spojená s cenou za její dosažení a příštím směrovačem (*next hop*) v cestě jsou uloženy do směrovací tabulky.

- Pokud nenastanou žádné změny, OSPF odesílá pravidelně LSU. (Interval odesílání je několika násobně delší než u *Hello* paketů.) Pokud nastanou změny v síti, jsou odeslány prostřednictví link-state paketů.

- Algoritmus klade každý směrovač do kořene stromu a provádí vlastní výpočet nejkratší cesty. Každý směrovač má tedy vlastní obraz topologie. Nicméně všechny cesty se budou směrovat stejným směrem, protože se shoduje link-state databáze.

3.2.1 OSPF cena

OSPF cena (*cost, metric*) je údaj, podle kterého směrovač určuje cestu paketu. Vzorec pro výpočet OSPF ceny, není dle RFC 2328 [13] určen.

Společnost Cisco pro výpočet ceny používá vzorec 3.1, kde je OSPF cena nepřímo závislá na šířce pásma (rychlosti linky).

$$cena = 100000000/rychlost\_linky\_v\_bps \tag{3.1}$$

Cena linek v síti se postupně sčítá. Pokud do jedné sítě vedou dvě různé cesty se stejnou cenou, pakety směřující do této sítě budou rovnoměrně rozdělovány mezi obě cesty.

3.3 OSPF oblasti a hraniční směrovače

OSPF rozděluje síť hierarchicky do oblastí (*areas*). Směrovače v rámci jedné oblasti mají stejnou link-state databázi a vyměňují si informace pomocí link-state aktualizací LSU. Oblast je specifická pro rozhraní směrovače. Směrovač, který spojuje oblasti (patří do více oblastí) se nazývá hraniční směrovač (*area border router*, ABR). Směrovač spojující protokol OSPF a jiné směrovací protokoly (např. RIP, BGP, IS-IS, EIGRP) se nazývá hraniční směrovač autonomního systému (*autonomous system border router*, ASBR). Na obrázku 3.1 je uveden příklad.

3.4 Link-state Advertisements (LSA) paket

Existují různé typy LSA paketů, které vysílají směrovače v různých pozicích.

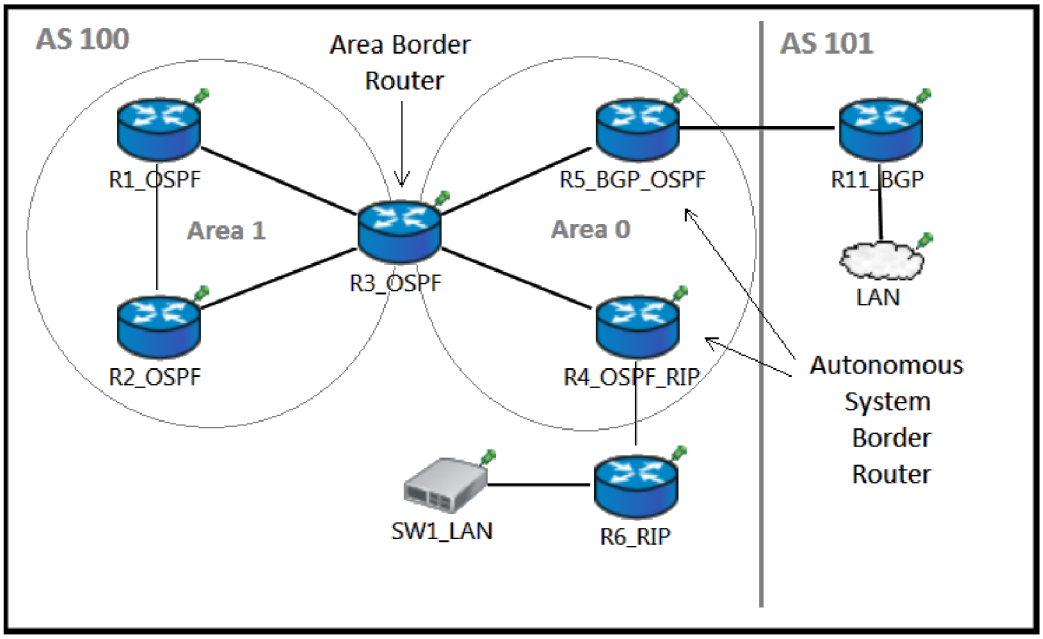
- ***Router Links*** - Popisují stavy a ceny linek (rozhraní) v rámci oblasti (Intra-area).

- ***Network Links*** - Pocházejí z designated směrovače (DR) a obsahují informace o linkách na Multi-access segmentu sítě.

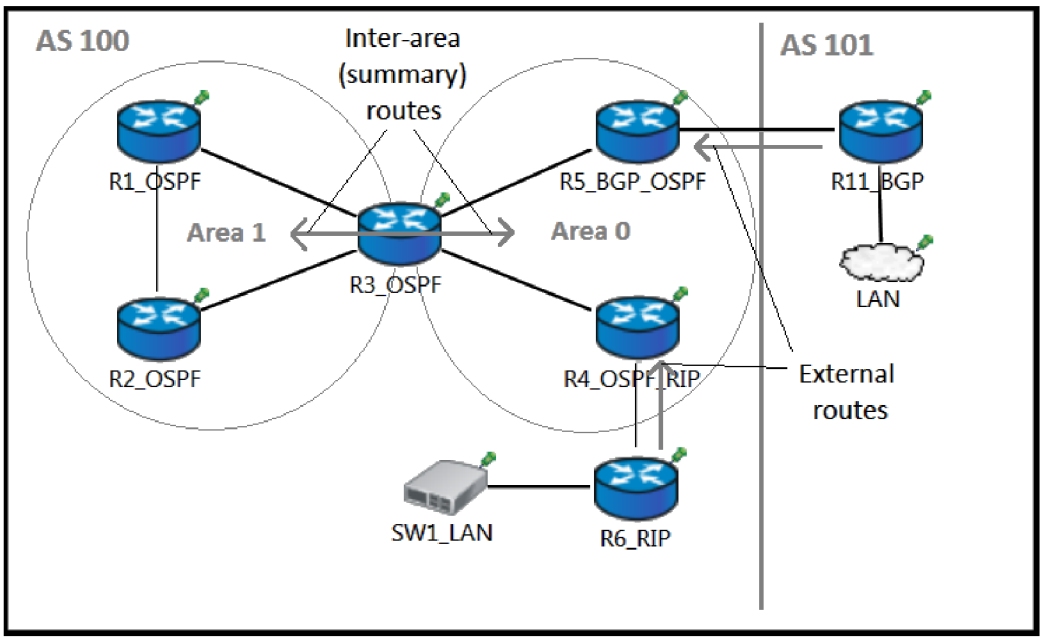
- ***Summary Links*** - Pocházejí z hraničního ABR směrovače šíří sumarizované informace z oblasti do jiné oblasti.

- ***External Links*** - Pocházejí z hraničního ASBR směrovače a obsahují informace z jiných protokolů nebo staticky zadané.

  Na obrázku 3.2 jsou zobrazeny názvy některých cest, které jsou předávány v link-state paketech.



Obrázek 3.1: Typy hraničních směrovačů



Obrázek 3.2: Typy cest, které obsahují link-state pakety.

3.5 Páteřní oblast (oblast 0)

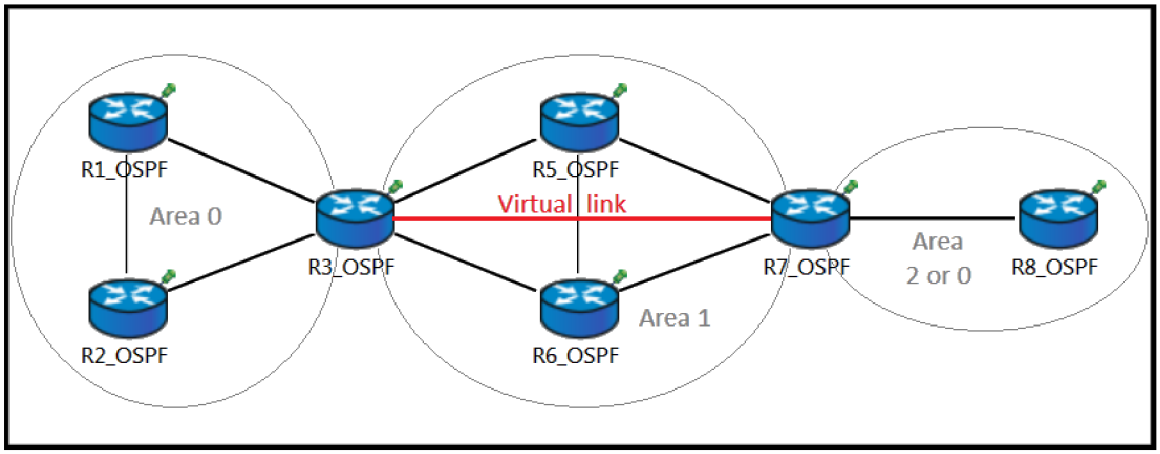
Pokud je v OSPF použito více oblastí, jedna z těchto oblastí musí být oblast 0 (páteřní, *backbone*). Do této oblasti musí být připojeny všechny ostatní oblasti. Ty předávají do páteřní své směrovací informace (pakety LSA *Summary links*) a páteřní předává informace ze všech oblastí do jednotlivých oblastí stejným způsobem.

3.6 Virtuální linka

Virtuální linka lze zkonstruovat prostorem jedné oblasti. Linka se vytváří mezi dvěma směrovači ABR, přičemž musí být alespoň jeden z nich připojen k páteřní oblasti. Používá se pouze ve výjimečných případech (např. dočasné zprovoznění sítě při poruše). Lze použít ve dvou případech:

- Připojení oblasti, která není fyzicky připojena k páteřní oblasti, přes jinou oblast, která je připojena páteřní oblasti.
- Propojení nespojité páteřní oblasti přes jinou oblast.

Na obrázku 3.3 je vidět příklad virtuální linky. Pokud bude pravá oblast 0, propojuje linka dvě části páteřní oblasti 0 a pokud bude např. 2, připojuje oblast k páteřní oblasti 0.



Obrázek 3.3: Virtuální linka

3.7 Sousedé (Neighbors)

Směrovače, které sdílejí segment sítě, jsou tedy sousedy na segmentu. Sousedství se navazuje pomocí *Hello* protokolu. Každý směrovač odesílá *Hello* pakety pomocí IP multikastu. Pokud směrovač obdrží *Hello* paket z jiného směrovače, začne ve svém *Hello* paketu odesílat informace o tomto směrovači. Když směrovač nalezne v cizím *Hello* paketu svou identifikaci, je sousedství navázáno.

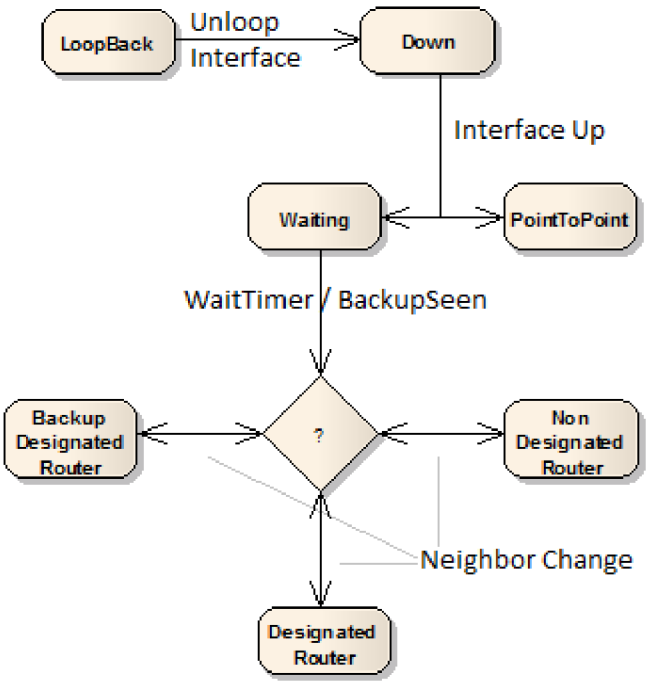
Sousedy se směrovače nemohou stát pokud:

- Na nich neběží stejná instance protokolu OSPF.
- Nejsou ve stejné oblasti.
- *Hello* interval není stejně dlouhý. Interval po kterém je odeslán další *Hello* paket.
- *Dead* interval není stejně dlouhý. Interval po kterém, pokud po jeho dobu nepřijde další *Hello* paket, je sousedství ztraceno.
- Příznak *Stub* oblasti není shodný. *Stub* oblast je vysvětlena dále v části kapitoly 3.10.

3.8 Sousedství (Adjacency)

Sousedství je dalším krokem po zjištění existence souseda. Směrovače po zjištění souseda, volí *designated* směrovač (DR) a záložní směrovač (*backup designated router*, BDR) na *multi-access* segmentu sítě (např. Ethernet). Centralizace předávání dat, směrovače nemusí komunikovat každý s každým (n x n spojení), ale komunikuje pouze jeden se všemi (n x 1 spojení), tím se šetří přenosové pásmo i rychlost konvergence OSPF protokolu. Pro zamezení výpadku DR směrovače, je stanoven BDR směrovač, který poslouchá a vytváří si stejnou databázi. Při výpadku DR se BDR stává DR a komunikuje s ostatními směrovači.

Na obrázku 3.4 jsou uvedeny stavy, kterými rozhraní prochází při ustanovení sousedství.



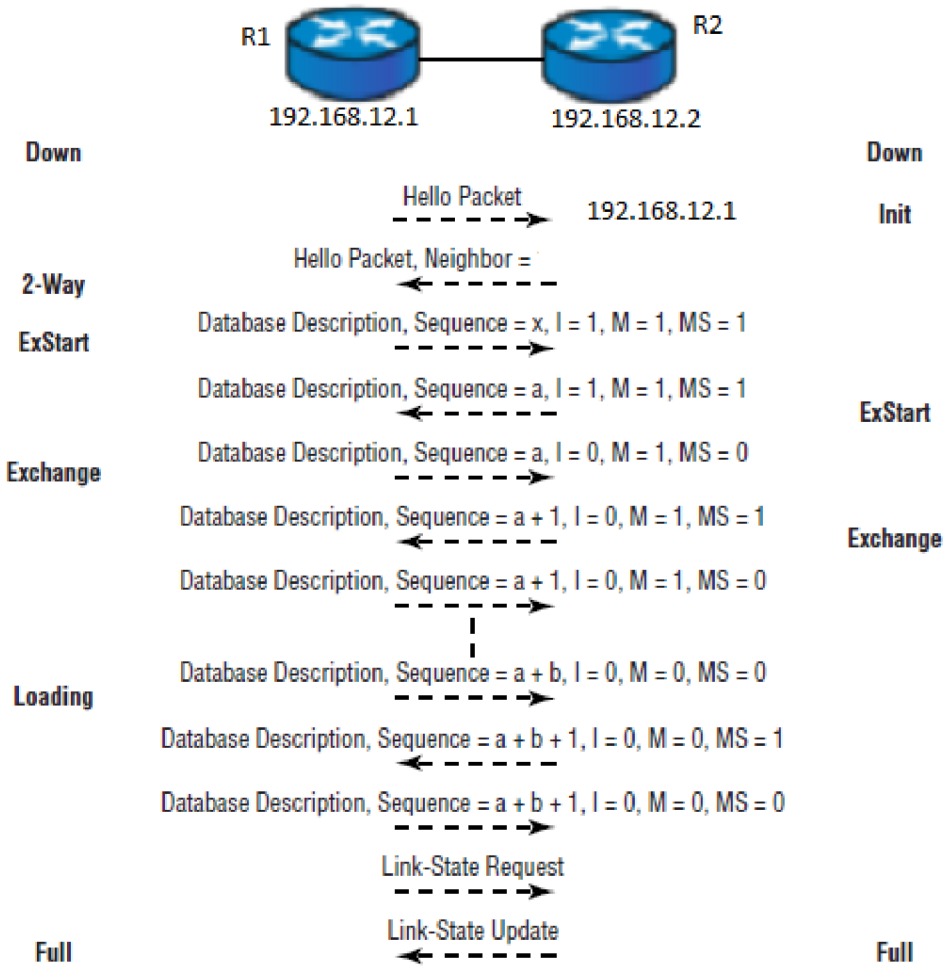
Obrázek 3.4: Stavy rozhraní

3.8.1 Výběr DR směrovače

Jako DR směrovač je nastaven směrovač s největší OSPF prioritou, při nerozhodném výsledku se určuje podle vyšší router-ID. BDR je určen směrovač s druhou největší prioritou (router-ID). Výchozí priorita je 1. V případě, že nechceme aby se směrovač mohl stát DR (BDR), nastavíme prioritu nula.

3.8.2 Ustanovení sousedství

Směrovače, které mezi sebou určují sousedství, prochází několika stavy. Po dokončení budou mít shodné link-state databáze. Na obrázku 3.5 je znázorněn sekvenční diagram stavů, při ustanovení sousedství.



Obrázek 3.5: Sekvenční diagram sousedství (obrázek přejat z [3])

- ***Down*** - Směrovač neobdržel žádné informace ze segmentu.
- ***Attempt*** - Na *non-broadcast multi-access* segmentech označuje, že od souseda nebyly přijaty nové informace. Pokouší se kontaktovat souseda zvýšením frekvence odesílání *Hello* paketů.

- ***Init*** - Rozhraní přicházejí informace od souseda, ale obousměrná komunikace nebyla stanovena.
- ***Two-way*** - Existuje obousměrná komunikace se sousedem (směrovač viděl sebe v *Hello* paketu). Volby DR, BDR byly provedeny. Na konci fáze se směrovač rozhoduje jestli pokračuje v ustanovování sousedství (DR) nebo ne v případě (*point-to-point* spojení).
- ***Exstart*** - Vytvářejí pořadového čísla pro zajištění aktuálních směrovacích informací.
- ***Exchange*** - Směrovač odešle svůj link-state databáze popisující paket (link-state database description paket).
- ***Loading*** - Směrovač si vybudoval link-state dotaz (*request*) na jemu zastaralé neúplné informace, odeslal jej a čeká na odpověď.
- ***Full*** - sousedství je stanoveno, informace vyměněny, směrovače mají stejné link-state databáze.

3.8.3 Sousedství na point-to-point segmentu

Směrovače budou tvořit vždy sousedství, nevolí DR, BDR.

3.8.4 Sousedství na non-broadcast multi-access segmentu

Na *multi-access* segmentu musí vždy DR (BDR) směrovač mít plnou konektivitu se všemi směrovači na segmentu. Směrovače, které takovouto konektivitu nemají, je třeba nastavit s OSPF prioritou nulovou, aby nemohly být DR (BDR) a nenastala tak chyba v konvergenci.

3.9 OSPF Sumarizace cest

Pro zmenšení objemu přenášených dat můžeme využít sumarizaci cest. Lze využít na hraničních směrovačích, můžeme místo několika sítí nastavit jednu cestu pro síť obsahující menší sítě. V OSPF můžeme sumarizovat při přechodu z oblasti do oblasti (*inter-area route sumarization*), nebo při zavádění externích cest (*external route sumarization*).

3.10 Stub oblast

OSPF umožňuje nakonfigurovat oblast jako *stub*, přičemž tato oblast nezná žádné informace o směro-vání v jiných oblastech a má nastavenou pouze výchozí bránu do páteřní oblasti. V případě propojení sítě s více směrovači do jiných oblastí, paket nemůže jít optimální cestou, protože musí projít přes odchozí směrovač stanoven výchozí bránou. Přes *stub* oblast nemůže být konfigurována virtuální linka. Všechny směrovače oblasti musí být konfigurovány jako *stub*, protože *Hello* paket obsahuje informace o *stub* oblasti, a pokud by informace nebyla stejná, směrovače nedokážou ustanovit sousedství.

Kapitola 4

Detaily OSPFv3 pro IPv6

Nová verze 3 protokolu OSPF vznikla pro IPv6 a je definována v RFC 5340 [7]. V OSPFv3 byla zachovaná většina algoritmů, ale některé změny musely být provedeny, kvůli jíné funkčnosti v IPv6 nebo jen kvůli délce adres v tomto protokolu, která se změnila ze 32 bitů na 128 bitů.

4.1 Linka

V IPv6 lze jednomu rozhraní přiřadit více adres IPv6, zatímco v IPv4 každé rozhraní mělo jednu adresu. Když v IPv4 komunikovaly zařízení přes médium po linkové vrstvě, musely být v jedné síti. V IPv6 mohou zařízení komunikovat ve více sítích (propojené rozhraní směrovačů jsou ve stejných sítích) přes médium nad linkovou vrstvou. Proto se označuje, že zařízení komunikují po lince (*per-link*), zatím co v IPv4 bylo označováno, že zařízení komunikují po podsíti (*per-subnet*).

4.2 Sémantika adresy

V OSPFv2 obsahovaly pakety adresové informace, v OSPFv3 jsou všechny adresové informace předávané LSA pakety. Pakety opouštějící směrovač jsou nezávislé na typu jádra protokolu (IPv4/IPv6).

- IPv6 adresy se objevují jen v *Link State Advertisements* (LSA) paketech.
- *Router-LSA* a *network-LSA* obsahují síťové adresy, pro zjednodušení vyjádření topologie sítě.
- OSPF *Router ID, Area ID, a LSA ID Link State* zůstávají na původní velikosti IPv4 (32 bitů).
- Sousední směrovače jsou vždy označovány *router ID*, v OSPFv2 byly označovány IPv4 adresou odchozího rozhraní.

4.3 Předávání LSA

V OSPFv3 jsou tři hranice předávání LSA.

- LSA je předávané na lokální lince, pro navázání nového spojení.

- LSA je předávané v rámci jedné oblasti (část směrovací domény).
- LSA je předávané do celé směrovací domény, předávání provádí směrovače na krajích oblastí (ASBR).

4.4 Více instancí na lince

V OSPFv3 lze jednu linku přiřadit do více spuštěných instancí OSPF. Tato podpora vznikla z důvodu potřeby překrytí jedné a více linek do více instancí, tyto linky pro jednotlivé instance jsou společné, ale ostatní části sítí se v obou instancích nesloučí.

4.5 Linkové adresy

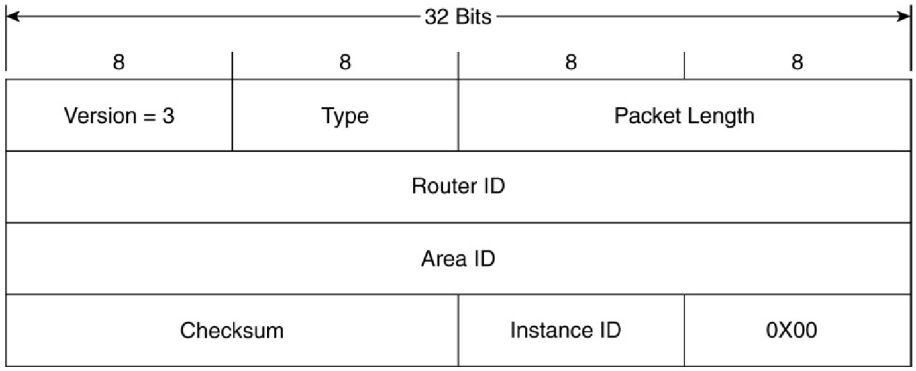
Rozhraní na jedné lince v IPv6 si automaticky konfigurují lokální linkovou adresu (*link-local address*). Tato adresa se použije pro navázaní spojení se sousedem. Linková adresa v IPv6 má prefix `FE80::/10` a není směrována na směrovačích. Směrovače při odesílání paketů nastavují jako zdrojovou adresu svou linkovou adresu na odchozím rozhraní. Výjimku tvoří virtuální rozhraní, která používají globální IPv6 adresu. Směrovač se učí linkové adresy všech ostatních směrovačů a používá je pro určení dalšího skoku.

4.6 Autentizace

V OSPFv3 byla odstraněna autentizace. Autentizaci zajišťuje protokol IPv6, pomocí k tomu určených rozšiřujících hlaviček.

4.7 Formát paketu

OSPFv3 běží nad protokolem IPv6. Všechna adresová sémantika byla odstraněna z OSPF hlavičky. Adresové informace jsou přenášeny pouze v LSA paketech. Hlavička paketu je zobrazena v následujícím obrázku 4.1.



Obrázek 4.1: hlavička OSPFv3 paketu (obrázek přejat z [4])

- Číslo verze protokolu bylo zvýšeno ze dvou na tři.
- V paketech *Hello* a *Database Description* bylo zvětšeno políčko volby (Option) na 24 bitů.
- Políčka autentizace a políčka typ autentizace byla odstraněna.
- *Hello* paket neobsahuje adresové informace. Paket obsahuje *interfaces-ID* (rozhraní ID), které jedinečně označuje směrovač na lince. *Interfaces-ID* je použito pro *link-state ID* v *network-LSA* pokud je směrovač *Designated Router* na lince.
- Byly přidány dva bity R-bit a V6-bit do políčka volby, které slouží k omezení směrování. V6-bit specializuje R-bit.
  - R-bit je nulový, OSPF směrovač může distribuovat topologii bez toho, aby předával tranzitní komunikaci.
  - V6-bit je nulový, OSPF směrovač může distribuovat topologii bez toho, aby předával IPv6 pakety.
  - R-bit je nastaven a V6 bit je nulový, OSPF směrovač nesmí předávat dále IPv6 pakety, ale může předávat pakety jiných protokolů.
- OSPFv3 paket obsahuje *Instance ID*, sloužící pro označení běhu více instancí OSPF na jedné lince.

4.8 Změny v LSA

V OSPFv3 neobsahují hlavičky OSPF v *router-LSA* a *network-LSA* žádné informace o směrování. Popisují tak síť více nezávislým způsobem. Byly přidány nové informace pro určení následujícího směrovače. Jména IPv4 LSA byla upravena tak aby odpovídala i jiným protokolům (IPv6).

- Pole Volby bylo odstraněno z OSPF hlavičky. Pole Volby bylo přidáno do těla *router-LSA, network-LSA, inter-area-router-LSA,* a *link-LSA*.
- V OSPF hlavičce bylo pole typ LSA rozšířeno na 16 bitů (použito místo po poli Volby). Nejvyšší tři bity určují rozsah přeposílání a zacházení s neznámými typy LSA.
- Adresy v LSA jsou vyjádřeny jako prefix sítě a délka prefixu (místo adresy a masky sítě). Výchozí brána je prefix s délkou nula.
- *Router-LSA* a *network-LSA* nemají žádné adresové informace, jsou tak protokolově nezávislé.
- Odesílající směrovač může informace z rozhraní rozdělit do více *router-LSA*. Přijímající směrovač příchozí *router-LSA* musí nejdříve pospojovat, a pak předat směrovači, který provede SPF výpočet.
- Nová zpráva LSA byla zavedena *link-LSA*. Zpráva je předávaná pouze na lince.
  - Informuje směrovač o linkových adresách ostatních směrovačů.
  - Informuje ostaní směrovače, jaké prefixy adres jsou přiřazeny na lince.

  - Umožňuje předávat informace o nastavení bitů Voleb pro linku.

- V IPv4 při odesílání *router-LSA* je použita IPv4 adresa dalšího rozhraní, tato adresa je ekvivalentní linkové adrese v IPv6 (není třeba využívat globální adresy, paket nepřekračuje linku). Při sestavování OSPF sousedství posíláním *Hello* paketů (zjištění linkové adresy souseda), může nastat problém v sítích *NBMA*, kde musí být ustaven DR, který dokáže navázat sousedství se všemi okolními směrovači.

- V poli volby LSA paketu je použita disjunkce tak, že každý směrovač odesílá *link-LSA*.

- Typ 3 *summary-LSA* je v OSPFv3 označován jako *inter-area-prefix-LSA*. A typ 4 *summary-LSA* je označován jako *inter-area-router-LSA*.

- *Link state ID* v *inter-area-prefix-LSA*, *inter-area-router-LSA*, *NSSA-LSA*, a *AS-external-LSA* již nepřenáší adresní informace. Slouží k identifikaci jednotlivých položek v *link-state-database*.

- V *network-LSA* a *link-LSA* nese *link-state ID* další význam. V těchto zprávách LSA *Link-state-ID* popisuje rozhraní směrovače. *Network-LSA* nese seznam všech směrovačů, které jsou připojeny na linku. *Link-LSA* přenáší seznam všech adres směrovačů připojených na linku.

- Novy typ LSA byl zaveden *intra-area-prefix-LSA*. Nese informace o všech IPv6 prefixech sítí, které jsou v IPv4 zahrnuty v *router-LSA* a *network-LSA*.

- Adresa pro přeposlání nebo externí směrovací označení je nyní přidáno v *AS-router-LSA* jako volba. Lze se odkazovat na další LSA pro přidání informací o cestě která není zahrnuta v OSPF.

4.9 Zacházení s neznámými typy LSA

Zavedením specifického chování směrovačů s neznámými typy LSA, je zaručena lepší přizpůsobivost protokolu. Pakety s neznámými typy LSA mohou být odeslány v linkovém rozsahu a nebo mohou být zpracovány a přeposlány dále, jako kdyby byl pochopen jejich obsah. K rozlišení zacházení paketu souží U-bit v hlavičce paketu. Tato vlastnost umožňuje propojit směrovače s plně implementovanou funkčností s jinými směrovači, které mají omezenou funkčnost.

Zavedení této vlastnosti ale umožňuje neřízený růst *link-state-database* v *stub* oblasti. OSPFv3 směrovače přeposílají do *stub* a *NSSA* oblastí zprávy LSA, jen když LSA typ je rozpoznán a U-bit nastaven na 1.

4.10 Podpora Stub/NSSA oblasti

V OSPFv2 byly tyto oblasti zavedeny pro snížení zpracovávaných informací ve velmi rozsáhlých sítí (zmenšení *link-state-database* a směrovacích tabulek). V OSPFv3 tato funkcionalita byla zachována. V OSPFv3 mohou chodit ve *stub* oblasti pouze zprávy *router-LSA*, *network-LSA*, *inter-area-prefix-LSA*, *link-LSA*, a *intra-area-prefix-LSA*. V *NSSA* oblasti je navíc zpráva *NSSA-LSA*.

4.11 Identifikace sousedů podle router ID

V OSPFv3 je pro identifikaci sousedů použito vždy *router ID* (označení formou IPv4 adresy). V OSPFv2 bylo také použito *router ID*, ale ne v případech na NBMA a point-to-multi-point sítích, kde se používala adresa rozhraních. Tato změna má vliv na příjem paketů, vyhledávání sousedů a příjem *Hello* paketů. *Router ID* 0.0.0.0 je vyhrazeno a nesmí se používat.

4.12 Potlačení link LSA

Pokud je tato vlastnost nastavena na rozhraní a linka není broadcast nebo NBMA, spojení *link-LSA* nevznikne.

Kapitola 5

OMNeT++ a INET

V této kapitole je popsáno vývojové a simulační prostředí OMNeT++ a framework INET, které slouží pro simulace v oblasti počítačových sítí. OMNeT++ je objektově orientovaný modulární diskrétní simulátor, který je šířen v podobě zdrojových kódů, čímž dosahuje přenositelnosti mezi systémy Windows, Linux a Mac OS. OMNeT++ je poskytován pro akademické a nekomerční potřeby zdarma, jinak je třeba využít komerční licenci.

5.1 OMNeT++

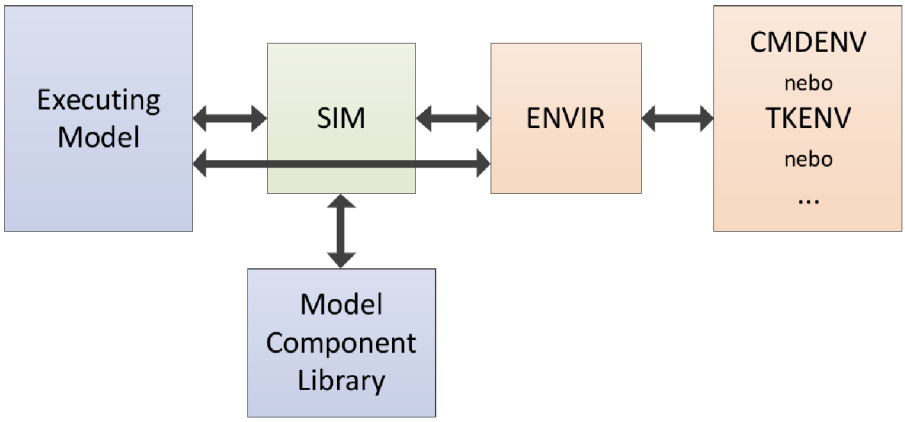
OMNeT++, jak již bylo uvedeno, je modulární simulátor, který využívá hierarchický systém modulů, které jsou propojeny zasíláním zpráv skrz brány. OMNeT++ je tedy diskrétní simulační prostředí, které má univerzální použití v případech, kde je třeba diskrétní simulace a zasílání zpráv mezi entitami. Lze tedy využít v simulaci telekomunikačních a počítačových sítí, validaci hardwarových architektur, vyhodnocování výkonnosti komplexních softwarových systémů, či dokonce k modelování různých obchodních procesů.

Aplikace OMNeT++ nabízí i vývojové prostředí OMNeT++ IDE, které je přizpůsobeno potřebám vývoje pro OMNeT++. Vývojové prostředí je postavené na volně dostupném vývojovém prostředí Eclipse. Kontroluje syntaxi jazyka C++, ve kterém je zapsáno chování modulů, ale také proprietární jazyk NED, který slouží pro popis jednotlivých modulů a jejich parametrů a zároveň pro popis jejich vzájemného propojení. OMNeT++ IDE také může jazyk NED reprezentovat graficky. Dále toto vývojové prostředí poskytuje funkce moderního vývojového prostředí správa projektů, kontrola a zvýrazňování syntaxe, inteligentní nápověda, vestavěný systém pro správu a verzování projektů, přímou vazbu na překladač a debugger.

Jak již bylo řečeno, simulace je sestavena z modulů v jazyce NED, který je graficky reprezentovatelný. Simulátor OMNeT++ může zobrazovat simulaci dle popisu v jazyce NED, a dále může zobrazovat běh zpráv mezi moduly. Při simulaci lze do jednotlivých modulů nahlížet a kontrolovat tak výměnu zpráv mezi moduly.

5.1.1 Simulátor OMNeT++

OMNeT++ má modulární architekturu. Obrázek 5.1 zobrazuje *"high-level"* pohled na celý simulátor. Jsou v něm popsány jednotlivé moduly simulátoru a vazby mezi těmito moduly.

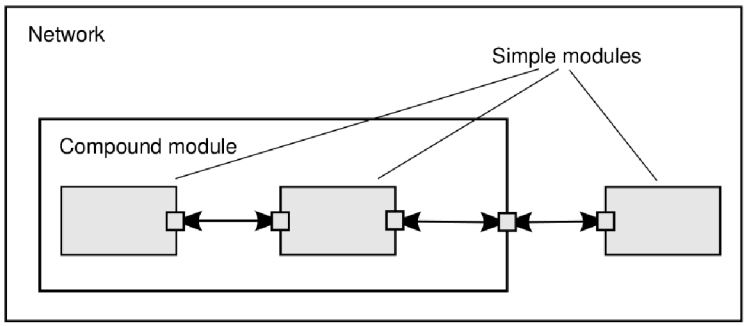


Obrázek 5.1: Složení modulů simulátoru [22] (obrázek přejat z [8])

- **Sim**
  Modul obsahuje simulační jádro a knihovny, které se používají při simulaci. Knihovny mohou být linkovány jak staticky, tak i dynamicky. Dále se také stará o funkci kalendáře událostí (generování, provádění a protokolování událostí).
- **Envir**
  Modul obsahuje hlavní smyčku programu (funkci `main()`). Dále prochází kalendář událostí a inicializuje jejich obsluhu. Zde se také načíná vstup simulace (konfigurační soubor `onetpp.ini`), zpracovává se zde výstup ze simulace a odchytávají výjimky. Výstup je vypisován prostřednictvím rozhraní `ev`.
- **Model Component Library**
  Modul obsahuje virtuální knihovnu všech zahrnutých modulů do simulace. Obsahuje podrobnou definici těchto zahrnutých modulů, dále pak jejich propojení a definici všech zpráv, kterými moduly komunikují.
- **Executing Model**
  Modul reprezentuje finální simulaci, obsahuje instance všech modulů z Model Component Library a také systémový modul, který je předkem všech modulů. Pomocí tohoto systémového modulu lze přistupovat do všech ostatních moulů.
- **CMDENV, TKENV**
  Posledním nepopsaným modulem simulátoru je uživatelské rozhraní pro přístup k simulaci a to textové `CMDENV`, či grafické `TKENV`. Každý z těchto prostředí má jiné využití. Pro rychlé zpracování několika simulací, které budeme analyzovat dle výstupů, je vhodnější použít textové rozhraní. Pokud potřebujeme vizualizaci problému a chceme do simulace zasahovat v jejím průběhu pak použijeme grafické rozhraní.

5.1.2 Jazyk NED

Jak již bylo uvedeno výše, pro popis simulace se využívá proprietární jazyk NED. V jazyku NED můžeme definovat vazby mezi moduly a také parametry modulů. V obrázku 5.2 jsou znázorněny tři typy modulů.



Obrázek 5.2: Typy modulů v simulátoru (obrázek přejat z [22] strana 21)

- **Jednoduchý modul**
  Jednoduchý modul je uveden klíčovým slovem `simple`. Tento modul je vázán na popis chování, které je zapsáno v jazyce C++. Modul může mít v jazyce NED zadané parametry, kromě konkrétních hodnot lze zvolit i ikona pro znázornění v simulaci. V modulu také můžeme definovat jednotlivé brány, které slouží pro propojení a komunikaci s jinými modely.

- **Složený modul**
  Složený modul je uveden klíčovým slovem `module`. Tento model nemá žádné chování, je pouze poskládán z dalších modulů, ať již jednoduchých či složených. V tomto modulu lze stejně jako v jednoduchém modelu definovat parametry a brány, ale také ještě moduly, ze kterých je poskládán, a také propojení mezi bránami modulů, z kterých je poskládán. Ve složených modulech můžeme ještě navíc využívat dědičnost.

- **Síťový modul**
  Síťový modul je uveden klíčovým slovem `network`. V tomto modulu se definuje kompletní topologie pro simulaci. Je to tedy modul, který je všem ostatním nadřazen. Je opět poskládán z dílčích modulu a jejich spojení, také lze zadat parametry. V tomto modulu lze navíc ještě zadávat typy. Nový typ, lze definovat a používat jako v klasických programovacích jazycích. V počítačových sítích lze například využít pro specifikování chybovosti, rychlosti či zpoždění linky.

Příklad definice OSPFv3 směrovače, který rozšiřuje směrovač IPv6, je na obrázku 5.3.

5.2 INET Framework

INET Framework je postavený nad OMNeT++ a používá stejný koncept (moduly a zasílaní zpráv) [6]. Jedná se o balík pro simulace nad rodinou protokolů TCP/IP. Obsahuje implementaci protokolů IPv4, IPv6, TCP, UDP, ARP, OSPF a další aplikační protokoly. A také jsou zde modely linkové vrstvy PPP, Ethernet a 802.11. Dále obsahuje podporu bezdrátových a mobilních sítí, které jsou převzaty z Mobility Frameworku.

INET Framevork umožňuje snadné vytváření simulací. Stačí vzít zvolené síťové komponenty, které se navzájem propojí. Díky modulárnosti OMNeT++ je velmi snadné upravovat síťové komponenty, dle potřeby do nich doplňovat potřebné služby nebo je odebírat.

```
module OSPFv3Router extends inet.nodes.ipv6.Router6
{
    parameters:
        IPForward = true;
        @display("i=abstract/router");
    submodules:
        ospfv3:OSPFv3Routing\{
            @display("p=465,287");
        }
    conections:
        ospfv3.ipOut --> networkLayer.ospfIn;
        ospfv3.ipIn --> networkLayer.ospfOut;
}
```

Obrázek 5.3: Směrovač OSPFv3

5.3 Projekt ANSA

Projekt ANSA *Automated Network-wide Security Analysis* [5] na Fakultě informačních technologií Vysokého učení technického v Brně se zabývá možnostmi a rozšiřováním balíku INET. Konečným výsledkem tohoto projektu by měl být nástroj pro formální analýzu a verifikaci bezpečnosti sítě. K tomu slouží věrný model sítě a protokolů vytvářený, dle reálných sítí.

V rámci projektu ANSA vzniklo již několik nových protokolů, které rozšiřují možnosti původního balíku INET. Mezi nové moduly patří RIP [15], RIPng [20], podpora ACL [18], podpora IPv6 (dual-stack zařízení) [23], SNMP [17], TRILL [12], PIM [16] a techniky QoS [9]. Dále byl vylepšen konfigurátor, který zařízení nastavuje. Současně s touto prací vznikají moduly EIGRP, skupina FHRP protokolů (HSRP, VRRP, GLBP).

Kapitola 6

Současný stav

V této kapitole se zabývá stavem projektu ANSA a frameworku INET jejich možnosti pro pro využití pro vývoj modulu OSPFv3.

Pro vývoj práce je použit OMNeT++ v aktuální verzi 4.4.1. a Framework INET je použit ve verzi 2.2 ze srpna 2013. V současnosti již je vydán nový INET verze 2.3 z března 2014. Tento framework není použit, protože musí dojít k přenesení celého projektu ANSA na novou verzi. Tato nová verze nepřináší pro rozpracované práce na OSPFv3 a EIGRP, žádné vylepšení a nové možnosti. Jak již bylo uvedeno, vývojové prostředí OMNET++, framework INET i projekt ANSA jsou stále aktuální a vyvíjí se nové verze.

6.1 OSPF v ANSA (INET)

Ve frameworku INET se nachází hotový modul `OSPF` pro IPv4. Funkčností tohoto modulu se zabývala bakalářská práce Modelování směrovacích protokolů OSPF v simulátoru OMNeT++, Martina Danka [23]. O této práce již uběhlo několik let a modul byl původními tvůrci zdokonalen.

V projektu ANSA se v práci Modelování IPv6 v prostředí OMNeT++, Marek Černý [8] zabýval převážně podporou IPv6, ale také protokolem OSPFv3, ze kterého při jeho rozsáhlosti (cca přes 10 000 řádků kódu) zpracoval odesílání *Hello* paketů. Od této práce uběhlo také několik let, a IPv6 bylo v INET zdokonaleno, stejně jako předloha OSPF pro jeho `OSPF6` modul (OSPF pro IPv6).

6.2 Směrovací protokoly

Framework INET obsahuje směrovací protokoly: BGP, OSPV verze 2, který není stále úplně doladěný a obsahuje několik komentářů v místech, kterým by se bylo potřeba věnovat, a nové směrovací protokoly RIP, GPSR a DYMO, které jsou novinkou nového balíku INET 2.3.

V projektu ANSA byly vytvořeny další směrovací protokoly: RIP [15], RIPng [20], IS-IS [12] a paralelně vzniká EIGRP.

Pro INET existuje také knihovna INETMANET, která implementuje směrovací protokoly DSDV, AODV, DSR a DYMO.

6.3 IPv6

Aktuálností a obsahem IPv6 ve framevorku INET se zabývala práce Marka Černého [8], od této práce z roku 2011 již uplynul také nějaký čas a INET byl vylepšen, nicméně IPv6 podpora není stále doladěná a obsahuje také komentáře pro dokončení. Pro implementaci směrovacího protokolu OSPFv3 potřebujeme směrovací tabulku, data IPv6 na rozhraní směrovače a čtení dat z konfiguračního XML souboru.

6.3.1 Směrovací tabulka

Základní směrovací tabulka `RoutingTable6` pro IPv6 je již ve frameworku INET. Tato tabulka ovšem nepodporuje označení rozdílných směrovacích protokolů, lze do ní pouze zadávat administrativní vzdálenost. V projektu ANSA směrovací byla vytvořena lepší směrovací tabulka `ANSARoutingTable6`, do které lze zadávat k cestě, také protokol, ze kterého cesta pochází. Hledání v obou uvedených směrovacích tabulkách probíhá sekvenčně od nejdelšího prefixu. Směrovací tabulka neřeší cesty porovnání cest, takže v ní může existovat více cest do stejného cíle s různou cenou.

Přehled vybraných metod směrovací tabulky:

- `addRoutingProtocolRoute()` a `addStaticRoute()` - přidání cesty
- `getRoute()` a `getNumRoute()` - získání určité cesty
- `removeRoute()` - odstranění cesty
- `doLongestPrefixMatch()` - vybrání nejlepší cesty

Další metodou která je v práci využitá, je `receiveChangeNotification()`, která přijímá oznámení o událostech. Touto metodou se například dozvídá směrovací tabulka o změnách na rozhraní směrovače. Tato metoda je ovšem v součastnosti pouze kostrou, pro vypracování. Dynamické změny na konfiguraci rozhraních směrovací tabulka nezpracovává, a zná konfiguraci rozhraní pouze načtené z konfiguračního souboru.

Přístup (ukazatel) do směrovací tabulky se získává prostřednictvím `RoutingTable6Access`. Pomocí tohoto přístupu můžeme volat veřejné metody třídy (modulu `ANSARoutingTable6`), není tedy připojen prostřednictvím hradel a bran.

6.3.2 Cesta

Cesta je záznam směrovací tabulky `IPv6Route`, tato třída umožňuje zadávat parametry cesty. Vytvořená cesta je vkládaná do tabulky. Výše uvedené rozšíření o typ směrovacího protokolu je možné díky cestě `ANSAIPv6Route`, která dědí z původní cesty `IPv6Route` a přidává další parametry. Pro administrativní vzdálenost a označení protokolu jsou vytvořené konstanty příklady pro OSPF: pro vzdálenost `dOSPF` a pro protokol `pOSPFintra`, `pOSPFinter`, `pOSPFext1`, `pOSPFext2`, `pOSPFNSSAext1` a `pOSPFNSSAext2`.

Níže je uveden přehled vybraných metod cesty. Níže uvedené metody nastavovací (`set`) mají obdobnou metodu pro získání (`get`), níže uvedené metody uvedené pro získání, nemají obdobnou metodu nastavení, ale hodnoty se předávají přímo do konstruktoru (`get`).

- `getDestPrefix()` - získání prefixu
- `getPrefixLength()` - získání délky prefixu

- `setMetric()` - nastavení cesty
- `setInterfaceId()` - nastavení odchozího rozhraní
- `setAdminDist()` - nastavení administrativní vzdálenosti
- `setRoutingProtocolSource()` - nastavení zdrojového protokolu

6.3.3 Rozhraní směrovače

Rozhraní směrovače je třída obsahující všechny informace o rozhraní. Nese na sobě data pro IPv4 i IPv6. Další důležitou informací je identifikátor rozhraní `interfaceID`, díky kterému dokážeme identifikovat jednotlivé rozhraní směrovače.

IPv6 data na rozhraní (`IPv6InterfaceData`) nesou všechny přiřazené adresy a nad těmito daty jsou volané metody pro přiřazení multicastové adresy. Tyto metody mohou být funkční, ale bohužel není funkční směrovací tabulka, jak je uvedeno výše, a tyto změny se nezapíšou do směrovací tabulky, tudíž síťová vrstva dotazující se směrovací tabulky na cílovou adresu příchozího paketu, dostane odpověď o neznámé adrese a příchozí multicastový paket nemůže být přijat a zpracován. Toto nežádoucí chování můžeme obejít tím, že rozhraní, na kterých budeme chtít zapnout OSPFv3 směrování, nastavíme dané OSPFv3 multicastové adresy v konfiguračním souboru.

6.3.4 Čtení konfiguračního souboru

V OMNeT++ existují parametry pro nastavování modulů napsané v jazyce NED. Nicméně spravovat tyto parametry pro velmi rozsáhlé sítě je velmi nepraktické a neflexibilní. K řešení se nabízí využití členských funkcí jazyka C++, které načtou konfiguraci z lépe dostupných a přehledných souborů například formátu XML. Ve frameworku INET existují moduly pro načtení konfigurace např. pro IPv6 `FlatNetworkConfigurator6`. Nicméně tento přístup ve frameworku INET není jednotný a například modul protokolu OSPFv2 `OSPFRouting` si konfigurační soubor čte sám.

V tomto ohledu projekt ANSA postupuje jednotně a byl v něm vytvořen modul pro konfiguraci `deviceConfigurator`, který slouží pro konfiguraci, všech modulů co byly v rámci tohoto projektu vytvořeny. Jednotlivé moduly volají metody konfigurátoru a ten jim vrací naplněné struktury daty.

Kapitola 7

Návrh a implementace

V této kapitole je popsán návrh a implementace směrovače a modulu OSPF pro IPv6. Návrh vycházel již z hotových částí projektu INET, přesněji byl vytvořen tak, aby byl modulární kompatibilní v rámci projektu INET.

Nejdříve bude popsán samostatně směrovač, na kterém běží proces OSPF verze 3 pro IPv6, a v dalších částí kapitoly se budeme zabývat jednotlivými části směrovacího procesu, které jsou uvedeny v části kapitoly 7.2.

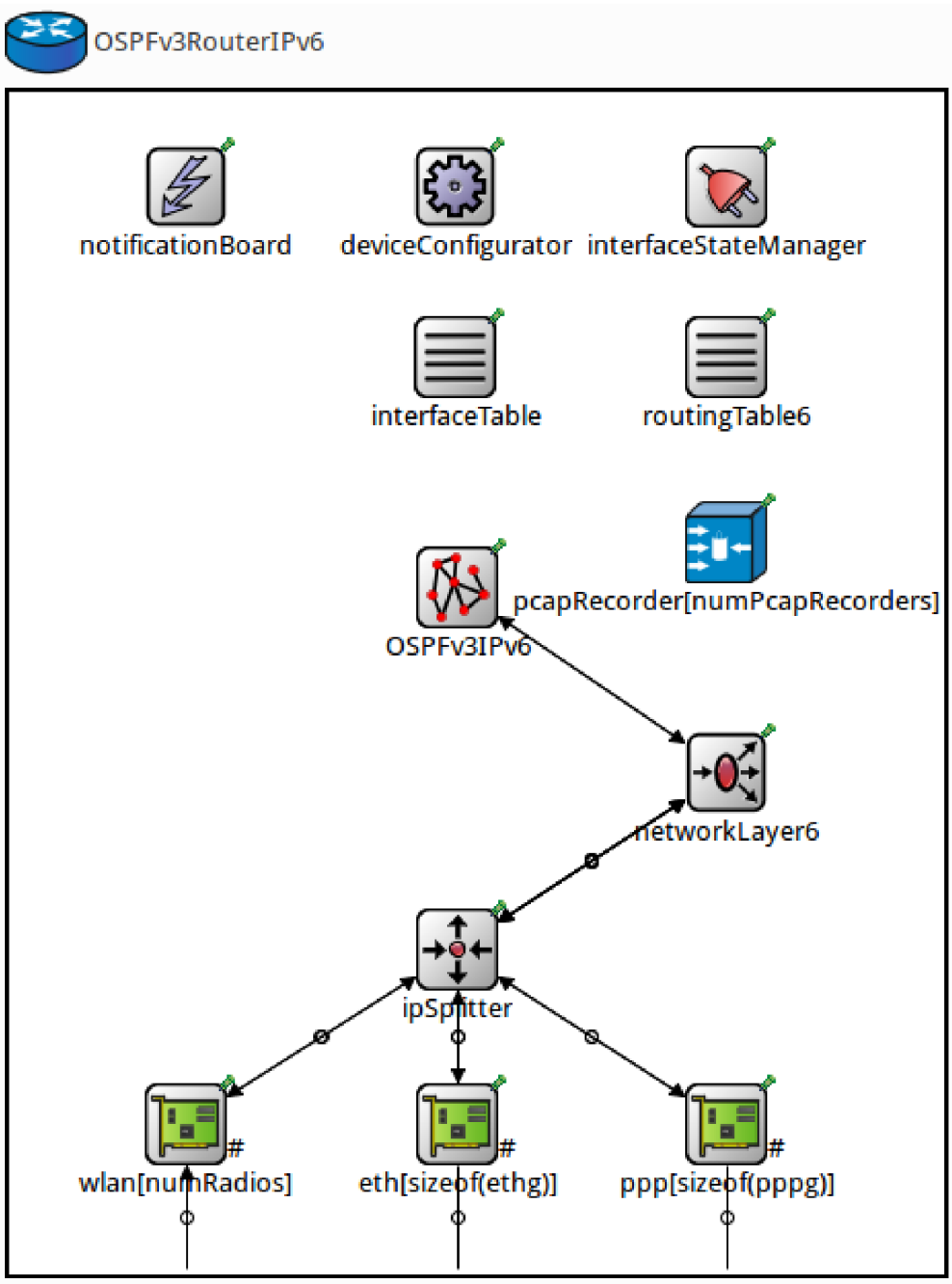
7.1 Směrovač OSPF

Sestavení směrovače OSPF verze 3 pro IPv6 je zobrazen na obrázku 7.1. Obrázek je přímé propojení v jazyce NED v simulačním prostředí OMNeT++, stejně jako následující obrázky v této kapitole.

Návrh vychází ze současného `ASNARouter`, který je *dualstack*, podporuje protokoly IPv4 a IPv6. V tomto směrovači pro OSPF verze 3 `OSPFv3RouterIPv6`, jsou vynechány implementované ostatní moduly v rámci ANSA (`RIP`, `RIPng`, `PIM`, `IS-IS`), směrovací protokol `OSPF`, síťová vrstva pro IPv4 `networkLayer`, směrovací tabulka `routingTable` a modul protokolu `UDP`.

Z původního `ASNARouter` zůstaly všechny rozhraní, `ipSplitter`, dále zde zůstaly nepostradatelné části jako síťová vrstva pro IPv6 `networkLayer6`, směrovací tabulka pro IPv6 `routingTable6` a tabulka rozhraní `interfaceTable`. Byly zde ponechány také moduly pro upozorňování na změny `notificationBoard`, manager změn rozhraní v průběhu simulace `interfaceStateManager` a modul pro úvodní nastavení (čtení konfiguračních souborů) `deviceConfigurator`, do kterého byly doplněny metody pro čtení konfigurace OSPFv3. `ipSplitter` byl ponechán ačkoli je určen pro rozdělení provozu IPv4 a IPv6, model nekomplikuje, naopak zjednodušuje pro budoucí doplnění směrovače. Prostor vlevo po IPv4 modulech byl stejně jako `ipSplitter` ponechán pro doplňování potřebných modulů.

Byl zde doplněn nový modul pro OSPF verze 3 IPv6 `OSPFv3IPv6`, kterým se budeme zabývat v dalších částech této kapitoly. Modul `OSPFv3IPv6` je připojen jedním hradlem, ale návrh počítá s tím, že bude moci být více procesů OSPFv3, které budou propojeny, každý svým hradlem, podobně jako jsou řešena připojení rozhraní.

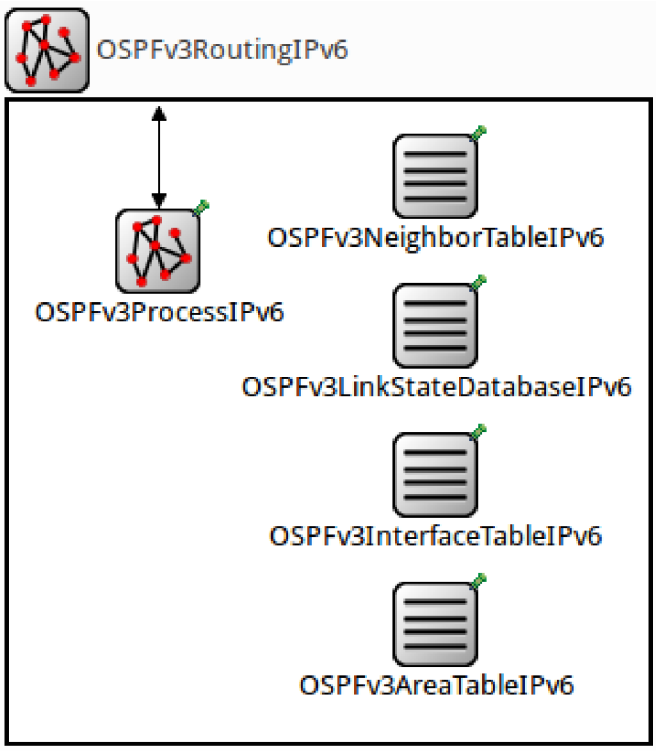


Obrázek 7.1: Směrovač OSPF IPv6

7.2 Modul OSPFv3 pro IPv6

V této části je popsán návrh modulu OSPFv3 pro IPv6 `OSPFv3RoutingIPv6`, který se připojuje k síťové vrstvě jak je uvedeno v minulé části kapitoly.

Tento modul je navržen podle novějších požadavků v rámci projektu ASNA na větší modularitu. Oproti staršímu RIPmg, který má směrovací proces implementovaný v rámci jednoduchého modelu, nově vyvíjené moduly OSPFv3 a EIGRP jsou rozděleny na proces a datové tabulky. Návrh modulu `OSPFv3RoutingIPv6` je zobrazen na obrázku 7.2.



Obrázek 7.2: Modul OSPFv3 IPv6

Tento modulární návrh umožňuje, v průběhu simulace, snadnou kontrolu dat obsažených v datových tabulkách, rozkliknutím modulu tabulky. V OSPF toto rozdělení trochu komplikuje implementaci, většina dat je přiřazena pod jinou datovou strukturu dle specifikace v RFC [7] (např. oblast, rozhraní, atd.). Například sousedé jsou uloženi pod rozhraním a většina LSA (např. *networkLSA*, *routerLSA* a *IntraAreaPrefixLSA*) náleží do oblasti. Tabulky se připojují na začátku do procesu. Při tomto návrhu vzniklo na procesu několik nových metod, připomínající dotazy na relační databáze s požadavky, podle kterých jsou vraceny jednotlivé záznamy v tabulce nebo části tabulek (datové struktury obsahující více záznamů). Tyto metody v procesu mají své odpovídající metody implementovány v tabulkách.

Implementace procesu je rozdělena do několika částí odpovídajících rozvržení zdrojových souborů do složek:

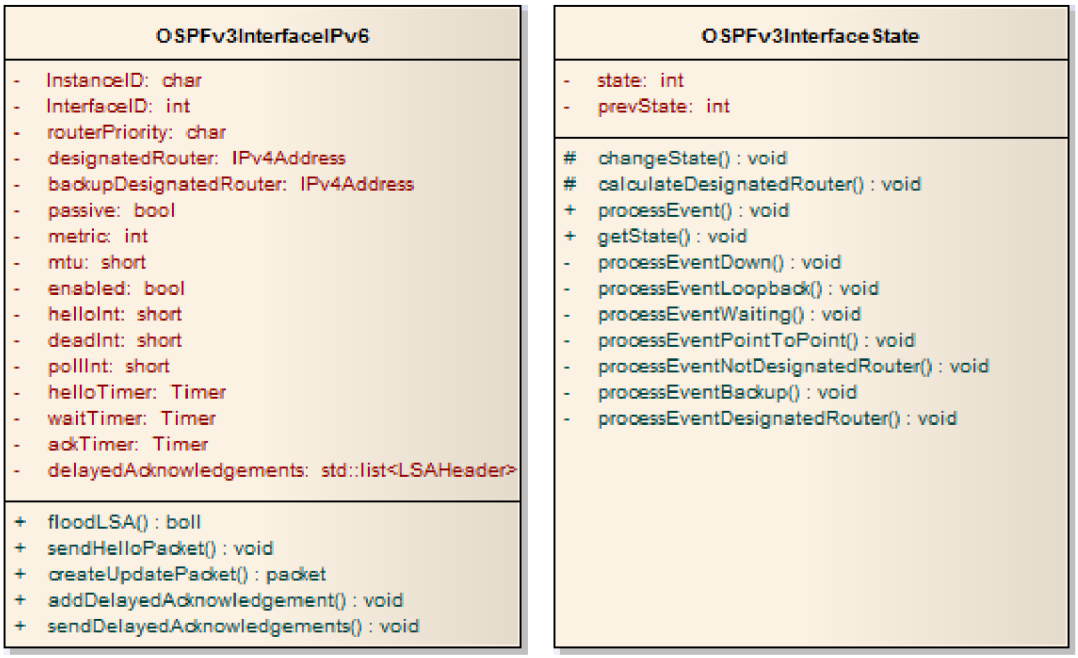
- `interface` - třídy pro nastavení a stavy rozhraní
- `nesagehandler` - třídy pro zpracování zpráv

- `neighbor` - třídy pro nastavení a stavy rozhraní
- `processIPv6` - třídy procesu a oblasti pro určení cest v síti
- `tables` - třídy pro tabulky zobrazené v návrhu
- nezařazeno - soubory jazyka NED pro směrovač a modul OSPFv3 a také třídy pro časovače a zprávy

Následující části této kapitoly odpovídají tomuto rozdělení.

7.3 Rozhraní - Interface

V této části práce jsou dvě třídy, třídu pro rozhraní `OSPFv3InterfaceIPv6` a třídu pro stavy rozhraní `OSPFv3InterfaceState`.



Obrázek 7.3: Přehled třídy `OSPFv3InterfaceIPv6` a třídy `OSPFv3InterfaceState`

Rozhraní

Třída pro rozhraní na sobě nese různé nastavení pro OSPFv3 jako parametry rozhraní, časovače, typ linky, stav rozhraní, intervaly pro přenos a časovače, ID instance, priorita, *Router ID* DR směrovače a BDR směrovače a nadřazená oblast. Jak je zobrazeno v obrázku 7.3 vlevo. Základní metody pro získávání a nastavování dat nejsou v obrázku zapsány.

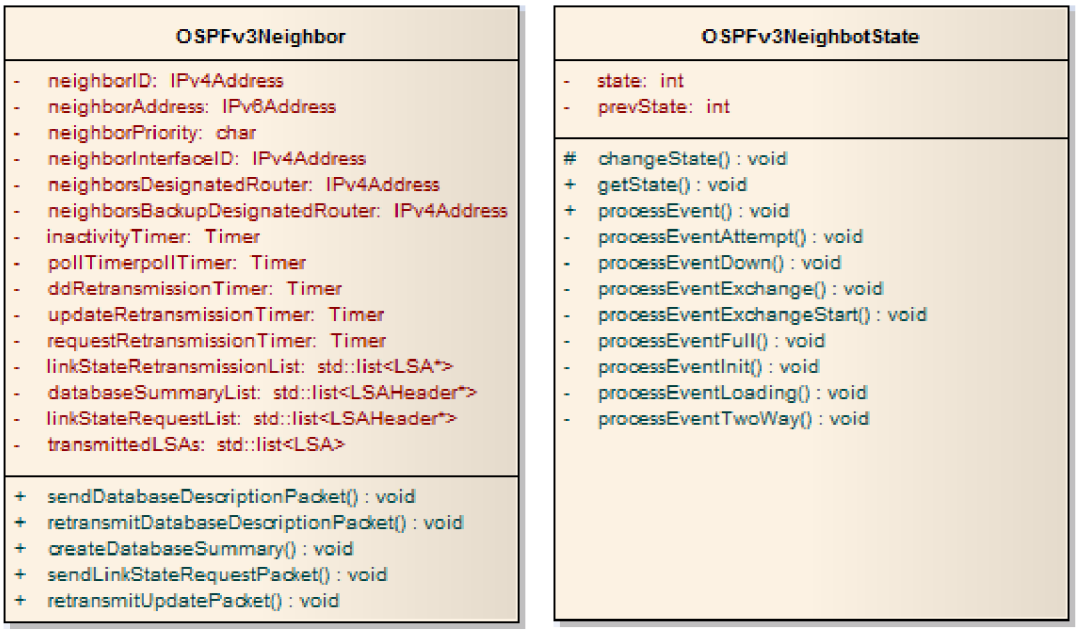
Mezi chybějící údaje proti specifikaci patří tabulka sousedů, která je přesunuta do centrální tabulky sousedů. Informace získává z centrální tabulky sousedů s identifikací dle ID. Další chybějící tabulkou, nově specifikovanou ve verzi 3, je tabulka, obsahující LSA typu 0x0008, tedy typu *LinkSLA*, nesoucí udaje o přilehlých IPv6 adresách na lince, je taktéž přesunuta do centrální tabulky obsahující LSA (`LinkStateDatabase`). Tato třída je podobná ekvivalentní třídě v modulu OSPF verze 2.

Stavy rozhraní

Třída pro stavy rozhraní nese na sobě stav rozhraní a zpracovává události (*event*), které přichází na rozhraní, podle nich nastavuje nový stav a případně vypočítává DR a BDR směrovač. Jak je zobrazeno v obrázku 7.3 v pravo. Tato třída je přepracovaná oproti modulu OSPF verze 2, protože použitý nevhodný návrh ve verzi 2, kdy se dle stavu vytvářela nová třída s přetíženou metodou pro zpracování události, způsobil vznik několika souborů s malým množstvím řádků kódu, což způsobilo nepřehlednost a vyšší výpočetní nároky na vytvoření nového objektu při každé změně stavu.

7.4 Soused - Neighbor

V této části práce jsou dvě třídy, třídu pro souseda `OSPFv3Neighbor` a třídu pro stav sousedství `OSPFv3NeighborState`.



Obrázek 7.4: Přehled třídy `OSPFv3Neighbor` a třídu `OSPFv3NeighborState`

Soused

Třída pro rozhraní na sobě nese různé nastavení a některé parametry, jako jsou na rozhraní, s tím rozdílem, že tyto údaje jsou sousedního směrovače. Mezi důležité data uložené na rozhraní přenesené LSA, dále LSA hlavičky pro databázi popisující paket a LSA hlavičky pro dotazovací paket. Jak je zobrazeno v obrázku 7.4 na levé straně. Základní metody pro získávání a nastavování dat nejsou v obrázku zapsány. Tato třída je podobná ekvivalentní třídě v modulu OSPF verze 2.

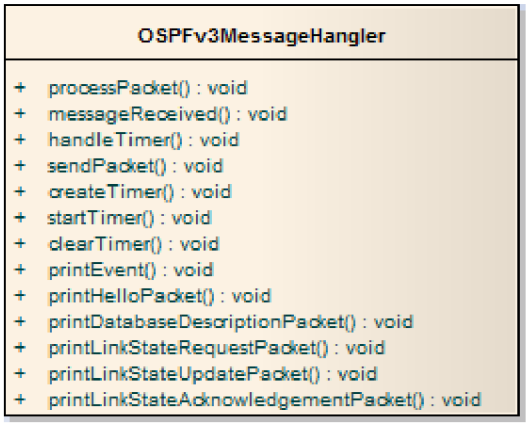
Stavy sousedství

Třída pro stavy rozhraní nese na sobě stav sousedství a zpracovává události (*event*), podle nich nastavuje nový stav. Třída je zobrazena v obrázku 7.4 na pravé straně. Tato třída je přepracovaná oproti modulu OSPF verze 2, stejně jako třída stavy rozhraní uvedená v minulé části kapitoly.

7.5 Zpracování zpráv - messageHandler

V této části jsou zařazeny příjemci zpráv, které je zpracují. Každý typ paketu má svou třídu na zpracování.

Veškeré zpracování příchozích zpráv a některých odchozích, které odpovídají spíš místu generování, na směrovači je přenecháno přídě `OSPFv3MessageHandler`. Tato třída třídí příchozí pakety a události. Události dále zpracovává a pakety předá třídě, která má na starosti daný typ paketů.



Obrázek 7.5: Přehled třídy `OSPFv3MessageHandler`

Třídy pro příjem paketů a událostí, každá tato třída má metodu na zpracování paketu `processPacket()`.

- `OSPFv3MessageHandler`
  Třída pro zpracování všech událostí a paketů předává události dále sousedovi a nebo rozhraní. Dále tato třída vypisuje informace o odchozích a příchozích paketech a příchozí pakety předává dle typu dalším třídám. Je znázorněna na obrázku 7.5.

- `OSPFv3HelloHandler` - Třída zpracovává inicializační paket (`HelloPacket`), dle něho nastavuje souseda a dále vyvolává události pro změnu stavů souseda.

- `OSPFv3DatabaseDescriptionHandler`
  Třída pro zpracování databáze popisující paket (`databaseDescriptionPacket`), třída také generuje události pro ovlivnění souseda. Kromě výše uvedené metody na zpracování paketu má též další metodu `processDDPacket()`, která zkoumá LSA hlavičky obsažené v paketu.

- `OSPFv3LinkStateRequestHandler`
  Třída zpracovává příchozí požadavky tvořené LSA hlavičky obsažené v příchozím

dotazovacím paketu (`requestPacket`), a pokud je LSA zpráva shodná s příchozí hlavičkou v databázi, tak ji odesílá zpět aktualizačním paketem.

- `OSPFv3LinkStateUpdateHandler`
  Třída zpracovává příchozí aktualizační paket (`updatePaket`), instaluje změny do databáze a metodou `acknowledgeLSA()` odesílá potvrzení o přijetí paketu.

- `OSPFv3LinkStateAcknowledgementHandler`
  Třída přijímá potvrzující pakety (`acknowledgementPacket`) a odstraňuje aktualizační pakety z sousedovi tabulky, který by je po určitém času znovu odeslal.

7.6 LSA

V této části jsou zařazeny LSA rozšiřující třídy, které umožňují k LSA třídám definovaným v souboru `OSPFv3Packet.msg` připojovat další data a metody.

V souboru `OSPFv3Packet.msg` jsou definované datové struktury OSPFv3 paketů a LSA, z této definice OMNeT++ generuje třídy, které nají základní metody pro získávání a nastavování definovaných dat a jsou připraveny pro odesílání po spojích zapsaných v souborech NED.

V této sekci k těmto generované třídy děděné do tříd, ve kterých jsou nesena další data jako další skoky v síti, cena uzlu a předchozí skok. Dále jsou zde metody pro porovnání a aktualizaci paketů a LSA.

7.7 Proces - processIPv6

V této sekci jsou tři třídy. První třída je `OSPFv3ProcessIPv6`, jde o hlavní třídu modulu OSPF verze 3 po IPv6, této třídě se věnuje následující část. Druhou třídou je třída oblasti `OSPFv3Area`. V této třídě se odehrává nejvíce směrovacích výpočtů. Poslední třída je `OSPFv3RoutingTableEntryIPv6`, tato třída rozšiřuje směrovací záznam (`ANSARouteIPv6`), tak aby k němu šly uložit dodatečné informace pro výpočet směrování.

7.7.1 Proces

Třída `OSPFv3ProcessIPv6` je, jak již bylo uvedeno výše, základní třídou modulu OSPF verze 3 pro IPv6. Tato třída se stará o inicializaci všech dat, přicházejí ji události a pakety, je připojená na datové tabulky celého směrovače (`interfaceTable`, `routingTable6`), na vlastní datové tabulky, uvedené v popisu modulu a budou ještě popsány v následující části, dále načítá konfiguraci pomocí `deviceConfigurator` a je připojen na síťovou vrstvu `networkLayer6`.

Inicalizace

K inicializaci modulů se v projektu ANSA, využívá `deviceConfigurator`, který v prvním kroku nastavuje hodnoty rozhraní, ve druhém směrovací tabulku a ve třetím již přijde na řadu směrovací protokol OSPF verze 3.

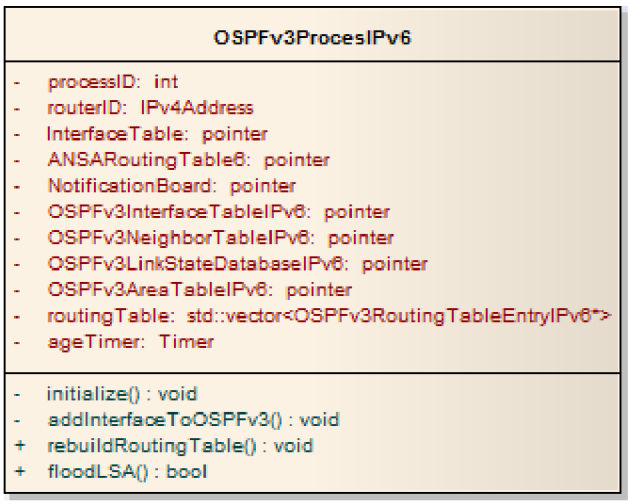
- V prvním kroku inicializace se proces propojí s datovými tabulky. Získá ukazatel na tabulku pomocí metody `ModuleAccess<datový typ>(název)`, kde datový typ je třída tabulky a název je její označení v jazyce NED. Hned poté nastaví ukazatel na

sebe, protože, jak již bylo zmíněno, datové tabulky nesou informace specifická pro různé třídy a někdy je třeba k rozhraní získat všechny sousedy, což lze díky volání metody procesu.

- Ve druhém kroku se proces spojí s datovými tabulky směrovače, směrovací tabulka a tabulka rozhraní a připojí se na informační kanál (`notificationBoard`), do kterého registruje události, o kterých chce být informován. Vzhledem k tomu, že tyto zprávy neposlouchá směrovací tabulka, zatím nebyly implementovány ani v modulu OSPFv3, a metoda `receiveChangeNotification()` pro příjem zprávy je pouze kostra.
- V dalším kroku se proces spojí s konfigurátorem a zavolá metodu pro nastavení dat `loadOSPFv3ConfigIPv6(this)`, kterému předává ukazatel na sebe, a konfigurátor nastavuje data, jak je uvedeno v části 7.10.
- Ve čtvrtém kroku spustí časovač, který s každou vteřinou simulačního času metodou `ageDatabase()`, zvýší staří záznamů v *link state* databázi.

Data a metody

Obrázek 7.6 znázorňuje přehled vybraných dat a metod třídy `OSPFv3ProcessIPv6`.



Obrázek 7.6: Přehled třídy `OSPFv3ProcessIPv6`

Třída procesu `OSPFv3ProcessIPv6` má na sobě zaznamenaná data, která slouží pro přístup do databází, dále také nese *router ID* a ID procesu, a třídu `messageHandler`, která automaticky vzniká při volání konstruktoru, takže je neoddělitelnou součástí a stará se o zprávy a události, jak bylo uvedeno v části 7.5. Dále nese kopii směrovacích cest, které jsou zapsány (pokud neexistují lepší cesty) do směrovací tabulky.

Proces by měl nést ještě data LSA pro externí směrování (ve všech oblastech stejné), které pochází z jiného zdroje. Tyto data stejně jako na rozhraní a oblasti přesunuta do *link state* databáze.

Tato třída má dvě přátelské třídy `OSPFv3Area` a `OSPFv3LinkStateDatabaseIPv6`, které mohou přistupovat k privátním datům procesu. Jsou využívány tak, že tyto třídy mají přístup přímo k metodám datových tabulek. Nemusí existovat metoda procesu. Důvody pro toto opatření jsou rozepsány v následující části 7.7.2.

Většina metod této třídy jsou krátké metody pro získávání dat z tabulek. Další metodou je metoda, která je určena pro konfiguraci, při čtení konfigurace je vytvořeno nové rozhraní (objekt `OSPFv3InterfaceIPv6`) a vložena na něj data. Toto rozhraní je předáno této metodě procesu `addInterfaceToOSPFv3()`, která přidá rozhraní do tabulky rozhraní v modulu a vyvolá startovací událost na rozhraní.

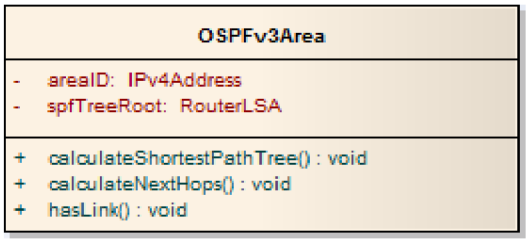
Nejdůležitější metodou je metoda přepočítávání směrovací tabulky (`rebuildRoutingTable()`). V této metodě je nejprve volání na všechny oblasti, které přepočítávají své vnitřní směrování, poté se počítá mezi oblastní směrování a nakonec externí směrování, při dokončování práce je hotovo pouze směřování uvnitř oblasti. V procesu se počítá s dalšími metodami pro výpočet externího směrování.

7.7.2 Oblast

Třída oblasti je důležitá pro směrování, odehrává se zde většina směrovacích výpočtů, všechny pro vnitřní oblast, a všechny pro mezi oblastní směrování se počítají na páteřní oblasti (*Backbone*).

Dle specifikace v RFC [7] by data LSA (*RouterLSA, NetworkLSA, InterAreaPrefixLSA, InterAreaRouterLSA* a *IntraAreaPrefixLSA*) a rozhraní měla být uložena v oblasti. Pro přiřazení metod co nejblíže datům, byly některé metody přesunuty z oblasti na tabulku rozhraní a *link state* databázi. Tímto však došlo rozdělení metod, které se volaly navzájem. Proto byla oblast a *link state* databáze přiřazeny k procesu jako přátelské třídy, aby měli k těmto metodám přístup.

Výběr metod a data, která zůstala na oblasti jsou znázorněna v obrázku 7.7. Oblast obsahuje významná data pouze ID oblasti a kořen stromu pro výpočet cest z tohoto směrovače. Dále obsahuje metody pro výpočet směrovacích cest v oblasti `calculateShortestPathTree()` a `calculateNextHops()`. Další metody pro výpočet cest mezi oblastmi nejsou zatím implementovány.



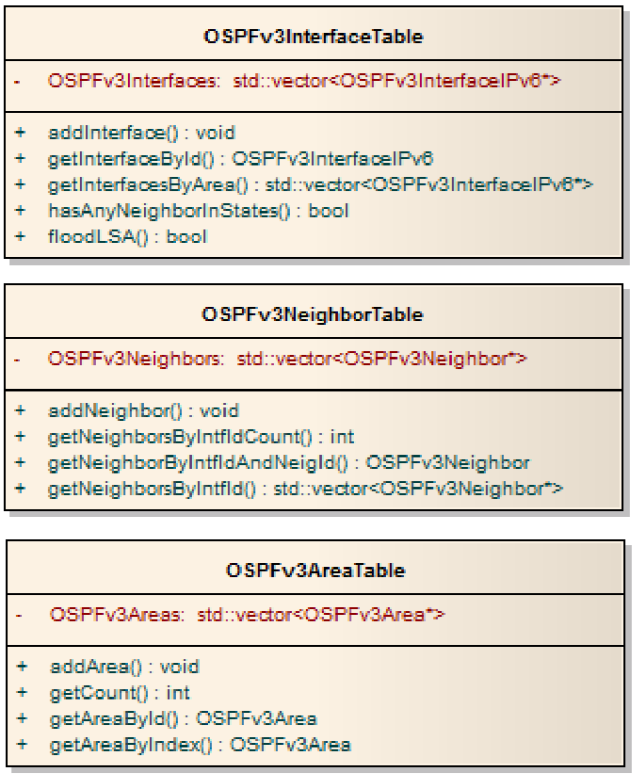
Obrázek 7.7: Přehled třídy `OSPFv3Area`

7.8 Tabulky - tables

Tabulky data a jejich metody jsou uvedeny na následujících obrázcích 7.8, výjimku uděláme u *link state* databáze, té se budeme věnovat níže.

Link State Databáze

Pro třídu `OSPFv3LinkStateDatabaseIPv6` představující *link state* databázi, jsou vytvořeny v hlavičkovém souboru jednotlivé záznamy pro každý typ LSA zvlášť, na těchto záznamech



Obrázek 7.8: Přehled tříd `OSPFv3AreaTable`, `OSPFv3InterfaceTable`, `OSPFv3NeighborTable`

jsou navíc údaje, které udávají původ záznamu, například jako oblast u *RouterLSA, NetworkLSA* a *IntraAreaPrefixLSA*.

Přehled třídy je zobrazen na obrázku 7.9. Mezi metody patří vytváření, vyhledávání a instalace LSA. Každý typ má vlastní metody. Další metoda je pro stárnutí databáze. Důležité jsou také operátory pro předání do proudu znaků (`operator<<`) pro jednotlivé záznamy. Díky tomu můžeme sledovat obsah tabulek při simulaci.

7.9 Nezařazeno

Mezi soubory, které nejsou zařazeny do složky patří definice v jazyce NED směrovvače a modulu OSPFv3. Dalšími soubory jsou automaticky generované souborů typu `.msg`, které charakterizují datové struktury, ze kterých OMNeT++ vygeneruje do tříd. Je tu soubor pro zprávy LSA a pakety OSPFv3 popsán 7.6 a soubor ve kterém jsou definovány všechny použité časovače.

7.10 Konfigurace

Pro konfiguraci byl upraven konfigurátor projektu ANSA `deviceConfigurator`, do kterého byly implementovány nové metody, Musel být také upraven `xmlParser`, který konfigurátoru rozděluje XML soubor.

Nové metody v konfigurátoru:

Obrázek 7.9: Přehled třídu `OSPFv3LinkStateDatabaseIPv6`

- `loadOSPFv3ConfigIPv6()`
  Tato metoda je volaná procesem, získá požadovaný element pro dané zařízení a volá následující metodu.
- `loadOSPFv3ProcessesConfig()`
  Metoda načítá data, která se vážou na celý proces, po načtení volá další metodu.
- `loadOSPFv3IPv6Areas()`
  Metoda postupně načítá data pro oblasti, po načtení volá poslední metodu pro rozhraní.
- `loadOSPFv3IPv6Interfaces()`
  Tato metoda postupně načítá nastavení pro všechny rozhraní pod jednou oblastí.

Přidané metody do `xmlParser`:

- `GetOSPFv3Process()` - Metoda vrací data ohraničené elementem "OSPFv3".
- `GetOSPFv3IPv6Area()` - Metoda vrací data ohraničené elementem "Area".
- `GetOSPFv3IPv6Interface()` -Metoda vrací data ohraničené elementem "Interface".

7.10.1 Formát dat v XML

Při návrhu uspořádání bylo vycházeno z objektových možností formátu XML. Data ve formátu mají jeden kořenový element "OSPFv3", pod kterým je možné konfigurovat procesy, oblasti a rozhraní. Každý tento element má své parametry a vyskytuje se ve vazbě 'jedna ku n' tak, že nadřazený element může mít více podřízených elementů.

Zde je rozpor s nastavením Cisco směrovačů, které v nastavení procesu můžou konfigurovat oblast a při konfiguraci rozhraní se zadává, do kterého procesu a oblasti náleží.

Při použití Cisco varianty přestává fungovat výhoda XML, protože se nedodržuje souvislý objekt a XML soubor ztrácí přehlednost.

Zápis v XML je znázorněn v tabulce 7.1, jsou zpracované pouze základní parametry. Parametry rozhraní se nemusejí zadávat. Výchozí hodnoty se v takovém případě volí následovně: typ rozhraní se zjistí podle fyzického rozhraní, `InstanceID` bude nastaveno na nula, priorita směrovače bude jedna, a rozhraní nebude pasivní.

```
<Routing6>
  <OSPFv3 ProcessID="1">
    <RouterID>
      <IPAddress>11.11.11.1</IPAddress>
    </RouterID>
    <Areas>
      <Area AreaID="0">
        <Interfaces>
          <Interface name="eth0">
            <Type>{Broadcast|PointToPoint|NBMA|PointToMiltiPoint}</Type>
            <InstanceID>{0-255}</InstanceID>
            <Priority>{0-255}</Priority>
            <Passive>{false|true}</Passive>
          </Interface>
        </Interfaces>
      </Area>
    </Areas>
  </OSPFv3>
</Routing6>
```

Tabulka 7.1: Příklad XML konfigurace

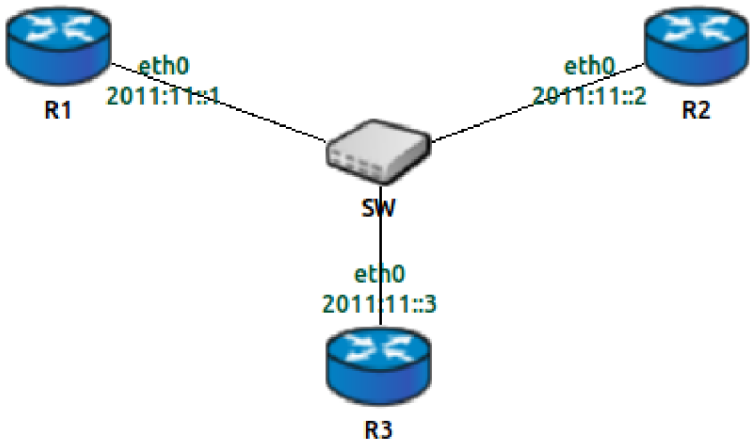
Kapitola 8

Testovací simulace

V této kapitole vyzkoušíme vytvořený vytvořený modul OSPF verze 3 pro IPv6. První dva testy jsou malého rozsahu a názorně ukazují, že je modul funkční. Třetí test je nejkomplikovanější, spojuje oba předchozí testy.

8.1 LAN

Pro první test jsou tři směrovače spojeny pomocí přepínače na jedné lince. Topologie je zobrazena na obrázku 8.1.



Obrázek 8.1: Testovací síť LAN (test-1)

Tímto testem se má otestovat výběr DR a BDR směrovače, dále přenos mezi DR směrovačem a ostatními směrovači, generovány zprávy `RouterLSA, NetworkLSA` a `Intra-AreaPrefixLSA`, poslední věcí, kterou jsem tímto testem chtěl otestovat byla rychlost volby DR směrovače.

Výsledky testu

Jak je vidět na obrázku 8.2 databáze obsahuje všechny tři typy zpráv LSA. Obsahuje tři zprávy `RouterLSA`, které označují přilehlou linku *Router ID* DR směrovače (10.0.0.3). Obsahuje také `NetworkLSA` s jednou zprávou pocházející z DR směrovače a *linkStateID* hodnotou

Obrázek 8.2: Obsah databáze na směrovači R3 (test-1)

rozhraní směrovače. Zpráva `NetworkLSA` obsahuje všechny připojené směrovače na lince. K této zprávě je také vygenerována zpráva `IntraAreaPrefixLSA`, která nese *linkStateID* a *advertisementRouterID* původní zprávy, a také prefix a délku prefixu na lince, pro kterou je generována.

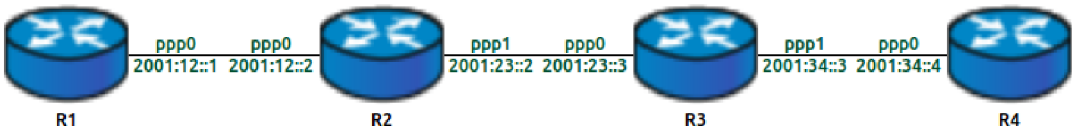
První ze zpráv `IntraAreaPrefixLSA`, je prázdná, byla vygenerována, v době kdy rozhraní bylo ve stavu čekání, v tu dobu obsahovala prefix tohoto rozhraní. Rozhraní ve stavu čekání patří do zpráv `IntraAreaPrefixLSA` stejným způsobem, jako rozhraní ve stavu *loopback*. Dále zde vidíme dvě prázdné zprávy `IntraAreaPrefixLSA`, ze směrovačů R1 a R2, které tím ukazují, že nemají navázané jiné linky než přes tuto linku. Tato linka je přenesena zprávou podle zprávy `NetworkLSA`.

Doba ustanovení směrovače DR a BDR je pět výměn *Hello* zpráv (přibližně 40 vteřin), při třetím testu, který probíhal v laboratoři jsem si vyzkoušel změřit ustanovení DR směrovače. A to tak, že ve schématu třetího testu jsem restartoval směrovače R2 a R3. Komunikaci jsem sledoval duplikací portu přepínače, jak je popsáno níže. Komunikace je na přiloženém

CD v souboru fast_switch.pcap. Dva směrovače odesílající pakety se objevily v čase 175 vteřin (od začátku sledování), první *Hello* zpráva s určeným směrovačem přišla v čase 193 vteřiny a okamžitě bylo na ní oběma směrovači odpovězeno. Doba ustanovení trvá pouze tři výměny (necelých 20 vteřin), je tedy zde prostor pro ladění modulu OSPFv3.

8.2 Směrování PPP

Tento test měl vyzkoušet směrování na základě `RouterLSA` a `IntraAreaPrefixLSA`, topologie sítě je velmi jednoduchá obsahuje čtyři směrovače v řadě, je znázorněna na obrázku 8.3.



Obrázek 8.3: Testovací síť PPP (test-2)

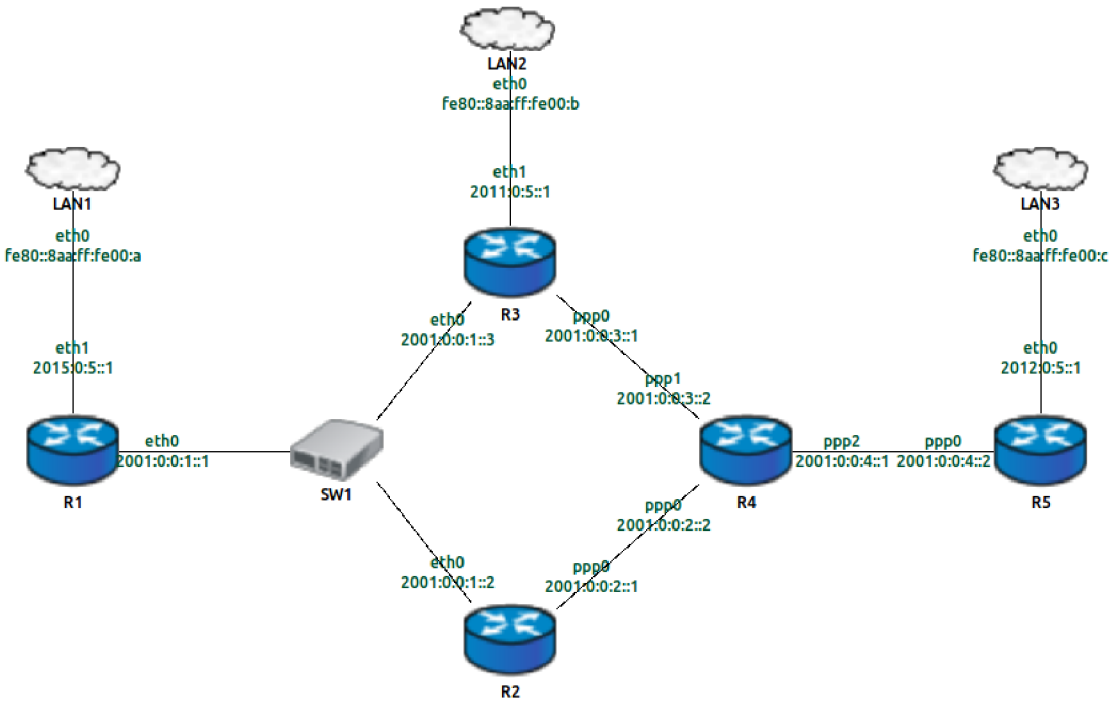
Výsledky testu

Data na směrovači R4, jak je vidět na obrázku 8.4, jsou kompletní. Pro čtyři směrovače jsou čtyři `RouterLSA` a čtyři `IntraAreaPrefixLSA`, z nichž směrovače uprostřed sítě mají dvě linky k sousedů, což přesně odpovídá topologii, správný přenos podporují i záznamy ve směrovací tabulce, přidané protokolem OSPFv3 (označené 'O').

Obrázek 8.4: Testovací síť PPP (test-2)

8.3 Směrování v oblasti

Testování s topologií na obrázku 8.5 bylo prováděno na reálné topologii v laboratoři. Pro testování jsem využil směrovače firmy Cisco modely řady 2811 s operačním systémem IOS verze 15.1(4)M7. Topologie byla stejná se stejnými IP adresami jako je na obrázku, jen místo připojených sítí byly využité rozhraní *loopback* nastaveny s IP adresou 201x:0:5::1/56.



Obrázek 8.5: Testovací síť pro směrování v oblasti (test-3)

Protože příkazy debug na Cisco směrovačích rychle mizí, při testování v laboratoři byl připojen na port přepínače počítač, a komunikaci na lince mezi směrovačem R1 byla replikována na port s připojeným počítačem.

Ke konfiguraci zařízení jsem využil příkazy v tabulce 8.1, poslední dva příkazy jsou zadané do přepínače pro replikaci provozu.

Výsledky testu

V tomto závěrečném a největším testu se nepodařila prokázat správná funkčnost modulu, byly doplněné jen některé směrovací tabulky a ne všemi inzerovanými prefixy.

V prvním testu jsem ověřil funkčnost modulu na *broadcast* lince, ke které byly připojeny tři linky, tím byl vyzkoušen přenos zpráv LSA. V tomto třetím testu je již potřeba zapojit přenesené zprávy do směrování, tudíž to vypadá na chybu při vytváření stromu směrování. OSPF verze dva toto směrování provádí, přímo podle zpráv NetworkLSA, které obsahují informace o síti a masce.

Tento test je tedy neúspěšný a vnitřní směrování v síti není plnohodnotné. Pro správné fungování modulu OSPFv3 je třeba dále upravovat a testovat.

```
R3#() ipv6 unicast-routing
R3#() ipv6 router ospf 1
R3#(-)  router id 10.0.0.3
R3#() interface Loopback 0
R3#(-)  ipv6 neable
R3#(-)  ipv6 address 2011:0:0::1/56
R3#(-)  ipv6 router ospf 1 area 0
R3#(-)  no shutdown
R3#() interface Serial 0/0/0
R3#(-)  clock-rate 64000

SW#() monitor session 1 source interface FastEthernet 0/1
SW#() monitor session 1 destination interface FastEthernet 0/24
```

Tabulka 8.1: Příkazy pro konfiguraci Cisco zařízení (test-3)

V posledním obrázku 8.6 je zachycená komunikace na lince mezi směrovačem R1 a přepínačem. Tato komunikace je uložená na přiloženém DVD ve formátu `.pcap` lze ji otevřít například programem Wireshark. Při prohlížení je dobré aplikovat filry, pro zobrazení stejné komunikace jako je v obrázku 8.6 zadáme filtr "OSPF".

134	166.315975	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	94 Hello Packet
139	166.328826	fe80::21f:caff:fe05:27b0	fe80::21f:caff:fe05:28d0	OSPF	94 Hello Packet
140	166.332745	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	82 DB Description
157	170.848432	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	82 DB Description
158	170.894619	fe80::21f:caff:fe05:27b0	fe80::21f:caff:fe05:28d0	OSPF	82 DB Description
159	170.896348	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	242 DB Description
160	170.897347	fe80::21f:caff:fe05:27b0	fe80::21f:caff:fe05:28d0	OSPF	142 DB Description
161	170.899084	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	94 LS Request
162	170.899091	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	82 DB Description
163	170.900247	fe80::21f:caff:fe05:27b0	fe80::21f:caff:fe05:28d0	OSPF	154 LS Update
164	170.900580	fe80::21f:caff:fe05:27b0	fe80::21f:caff:fe05:28d0	OSPF	154 LS Request
165	170.905458	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:27b0	OSPF	406 LS Update
166	170.938850	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	154 LS Update
170	171.437657	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	242 LS Update
171	171.738637	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	166 LS Update
174	172.064882	fe80::21f:caff:fe05:c3b0	ff02::5	OSPF	90 Hello Packet
175	172.066110	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	98 Hello Packet
180	172.077316	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	82 DB Description
181	172.080890	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	82 DB Description
182	172.080898	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	262 DB Description
183	172.082386	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	102 DB Description
184	172.084180	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	82 LS Request
185	172.084186	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	82 DB Description
186	172.086222	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	130 LS Update
187	172.120637	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	130 LS Update
188	172.121580	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	90 LS Acknowledge
189	172.488939	fe80::21f:caff:fe05:c3b0	ff02::6	OSPF	114 LS Update
190	172.524748	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	114 LS Update
191	172.535848	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	98 Hello Packet
195	173.405100	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	190 LS Acknowledge
196	173.406786	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	330 LS Acknowledge
198	174.580734	fe80::21f:caff:fe05:c3b0	ff02::6	OSPF	230 LS Acknowledge
199	175.430562	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	98 Hello Packet
200	175.942695	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	118 LS Update
202	176.432692	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	110 LS Update
203	176.832471	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	110 LS Update
204	176.833386	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	90 LS Acknowledge
210	178.444255	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	90 LS Acknowledge
218	178.930641	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	90 LS Acknowledge
219	178.932640	fe80::21f:caff:fe05:c3b0	ff02::6	OSPF	90 LS Acknowledge
227	180.780595	fe80::21f:caff:fe05:28d0	fe80::21f:caff:fe05:c3b0	OSPF	118 LS Update
228	180.781546	fe80::21f:caff:fe05:c3b0	fe80::21f:caff:fe05:28d0	OSPF	90 LS Acknowledge
230	181.268688	fe80::21f:caff:fe05:c3b0	ff02::5	OSPF	98 Hello Packet
233	182.267852	fe80::21f:caff:fe05:28d0	ff02::5	OSPF	98 Hello Packet
238	184.634643	fe80::21f:caff:fe05:27b0	ff02::5	OSPF	98 Hello Packet

Obrázek 8.6: Zachycená komunikace v laboratoři

Kapitola 9

Závěr

Tato práce se zabývá detaily protokolu OSPF, především jeho verzí pro IPv6. Je zde popsáno použité názvosloví, jsou zde uvedeny principy protokolu a jsou podrobně popsány změny, které v protokolu OSPF nastaly pro IPv6 oproti IPv4.

V rámci této práce vzniká rozšíření pro framework INET, který je použitelný v simulačním prostředí OMNeT++. S použitím vzoru, který poskytuje modul OSPF pro IPv4, podle kterého je vytvořen modul OSPFv3 pro IPv6. Modul OSPFv3 je rozčleněn do více modulů, na kterých je možno při běhu simulace sledovat stav a hodnoty vycházející v simulaci.

9.1 Dosažené výsledky

V práci se podařilo implementovat OSPFv3 modul pro IPv6 `OSPFv3ProcessIPv6` s omezenou funkčností. Tento modul neúplně zpracovává směrování v rámci jedné oblasti, přesto je připraven na implementaci většího rozsahu směrování. Hotové jsou rozhraní směrovače i s přechodem stavů, soused se stavy sousedství. Zcela je implementován příjem všech paketů. Plně jsou připraveny všechny druhy paketů i zpráv LSA. Plně jsou fungující datové tabulky pro sousedy, rozhraní a oblasti. Link state databáze funguje pro používané typy, má připravené tabulky pro všechny typy LSA, nejsou však implementovány metody pro jejich vkládání a vyhledávání, avšak tyto metody budou podobné stávajícím metodám.

9.2 Další možnosti práce

Tuto práci je možné dále rozšířit o směrování mezi oblastmi a směrování s externími zdroji. Tento protokol OSPFv3, jak je výše v jeho popisu uvedeno, je protokolově nezávislý, od konce roku 2012 se začal využívat pro směrování IPv4, tím se otvírají další možnosti vývoje tohoto modulu.

Literatura

[1] OSPF Design Guide; Cisco Systems, Inc. http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094e9e.shtml, 2005-08-10 [cit. 2014-05-25].

[2] IPv6 Multicast Address Space Registry. http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml, 2014-03-20 [cit. 2014-05-25].

[3] http://1.bp.blogspot.com/--CPln7e6JvI/UWgJMC_0FBI/AAAAAAAAAQc/4gISWmv7Stg/s1600/osp [cit. 2014-05-25].

[4] http://packetpushers.net/wp-content/uploads/2012/10/Deescription.png, [cit. 2014-05-25].

[5] ANSA Projekt. nes.fit.vutbr.cz/ansa/pmwiki.php, [cit. 2014-05-25].

[6] INET Framework. http://inet.omnetpp.org, [cit. 2014-05-25].

[7] Coltun, R.; Ferguson, D.; Moy, J.; aj.: OSPF for IPv6, RFC 5340. July 2008.

[8] Danko, M.: *Modelování směrovacích protokolů OSPF v simulátoru OMNeT++.* bakalářská práce, Brno, FIT VUT v Brně, 2009.

[9] Danko, M.: *Modelování kvality služeb v počítačových sítích.* diplomová práce, Brno, FIT VUT v Brně, 2012.

[10] Deering, S.; Hinden, R.: Internet Protocol, Version 6 (IPv6) Specification, RFC 1883. December 1995.

[11] Hinden, R.; Deering, S.: IP Version 6 Addressing Architecture, RFC 4291. February 2006.

[12] Hrnčiřík, M.: *Modelování L2 protokolů zajišťujících bezesmyčkovost.* diplomová práce, Brno, FIT VUT v Brně, 2012.

[13] Moy, J.: OSPF version 2, RFC 2328. April 1998.

[14] Postel, J.: Internet Protocol, RFC 791. September 1981.

[15] Rybová, V.: *Modelování a simulace návrhových vzorů směrování v počítačových sítích.* bakalářská práce, Brno, FIT VUT v Brně, 2009.

[16] Rybová, V.: *Modelování multicastového směrování v prostředí OMNeT++*. diplomová práce, Brno, FIT VUT v Brně, 2012.

[17] Smejkal, J.: *Dynamický stav modelu OMNeT++ pomocí SNMP*. diplomová práce, Brno, FIT VUT v Brně, 2012.

[18] Suchomel, T.: *Rozšíření simulátoru OMNeT++ o filtrovací pravidla ACL*. bakalářská práce, Brno, FIT VUT v Brně, 2009.

[19] Thomson, S.; Narten, T.; Jinmei, T.: Internet Protocol, RFC 4862. September 2007.

[20] Trhlík, J.: *Modelování distance-vektor směrovacích protokolů*. diplomová práce, Brno, FIT VUT v Brně, 2013.

[21] Varga, A.: OMNeT++. http://omnetpp.org, 20011 [cit. 2014-05-25].

[22] Varga, A.: OMNeT++ User Manual; OpenSin Ltd. http://omnetpp.org/doc/omnetpp/manual/usman.html, 20014 [cit. 2014-05-25].

[23] Černý, M.: *Modelování IPv6 v prostředí OMNeT++*. diplomová práce, Brno, FIT VUT v Brně, 2011.

Příloha A

Obsah DVD

Datový nosič CD obsahuje:

- Text bakalářské práce ve formátu PDF.
- Text bakalářské práce ve formátu TEX.
- Obrázky použité při tvorbě textu.
- Datové soubory získané při testování.
- Projekt ANSA s vytvořenými zdrojovými kódy modulu.
- Tolologie a konfigurace pro testování.

Zdrojové kódy modulu a příklady pro testování jsou uloženy v souborové struktuře projektu ANSA popsané v následující příloze B.

Příloha B

Spuštění projektu

Pro spuštění projektu je třeba nainstalovat OMNeT++. Stáhnout lze ze stránek projektu [21]. Pro lepší fungování se doporučují mít před instalací nainstalované knihovny MPI, PCAP a Akaora.

Po otevření vývojového prostředí OMNet++, zvolíme soubor a importovat existující projek, je nutné mít data z přiloženého DVD.

Lepší způsob bude importovat projekt pomocí verzovacího systému GIT, který je také součástí vývojového prostředí. Zvolíme tedy importovat a metodu Git. Projekt je veřejně dostupný na adrese: https://github.com/kvetak/ANSA. Stažení projektu touto cestou má tu výhodu, že máte k dispozici nejnovější verzi, ale i starší verze, projektu ANSA.

Soubory vytvořené v projektu ANSA

```
src/ansa/networklayer/ospfv3/

OSPFv3RoutingIPv6.ned
OSPFv3Router.ned
OSPFv3PacketIPv6.msg
OSPFv3Timer.msg
interface/OSPFv3InterfaceIPv6.cc
interface/OSPFv3InterfaceIPv6.h
interface/OSPFv3InterfaceState.cc
interface/OSPFv3InterfaceState.h
LSA/OSPFv3RouterLSA.cc
LSA/OSPFv3NetworkLSA.cc
LSA/OSPFv3InterAreaPrefixLSA.cc
LSA/OSPFv3InterAreaRouterLSA.cc
LSA/OSPFv3ASExternalLSA.cc
LSA/OSPFv3LinkLSA.cc
LSA/OSPFv3IntraAreaPrefixLSA.cc
LSA/OSPFv3LSA.cc
LSA/OSPFv3LSA.h
messagehandler/OSPFv3DatabaseDescriptionHandler.cc
messagehandler/OSPFv3DatabaseDescriptionHandler.h
messagehandler/OSPFv3HelloHandler.cc
messagehandler/OSPFv3HelloHandler.h
messagehandler/OSPFv3LinkStateAcknowledgementHandler.cc
messagehandler/OSPFv3LinkStateAcknowledgementHandler.h
messagehandler/OSPFv3LinkStateRequestHandler.cc
```

messagehandler/OSPFv3LinkStateRequestHandler.h
messagehandler/OSPFv3LinkStateUpdateHandler.cc
messagehandler/OSPFv3LinkStateUpdateHandler.h
messagehandler/OSPFv3MessageHandler.cc
messagehandler/OSPFv3MessageHandler.h
neighbor/OSPFv3Neighbor.cc
neighbor/OSPFv3Neighbor.h
neighbor/OSPFv3NeighborState.cc
neighbor/OSPFv3NeighborState.h
processIPv6/IOSPFv3Module.h
processIPv6/OSPFv3Area.cc
processIPv6/OSPFv3Area.h
processIPv6/OSPFv3ProcessIPv6.cc
processIPv6/OSPFv3ProcessIPv6.h
processIPv6/OSPFv3ProcessIPv6.ned
processIPv6/OSPFv3RoutingTableEntryIPv6.cc
processIPv6/OSPFv3RoutingTableEntryIPv6.h
tablesIPv6/OSPFv3AreaTableIPv6.h
tablesIPv6/OSPFv3AreaTableIPv6.h
tablesIPv6/OSPFv3AreaTableIPv6.h
tablesIPv6/OSPFv3InterfaceTableIPv6.h
tablesIPv6/OSPFv3InterfaceTableIPv6.h
tablesIPv6/OSPFv3InterfaceTableIPv6.h
tablesIPv6/OSPFv3LinkStateDatabaseIPv6.h
tablesIPv6/OSPFv3LinkStateDatabaseIPv6.h
tablesIPv6/OSPFv3LinkStateDatabaseIPv6.h
tablesIPv6/OSPFv3NeighborTableIPv6.h
tablesIPv6/OSPFv3NeighborTableIPv6.h
tablesIPv6/OSPFv3NeighborTableIPv6.h

examples/ansa/ospfv3/

DIP_LAN/dip_lan.ned
DIP_LAN/omnetpp.ini
DIP_LAN/config.xml
DIP_PPP/simple_test.ned
DIP_PPP/omnetpp.ini
DIP_PPP/config.xml
DIP_example/DIP_example.ned
DIP_example/omnetpp.ini
DIP_example/config.xml

Soubory upravené v projektu ANSA

src/ansa/util/deviceConfigurator/

deviceConfigurator.cc
deviceConfigurator.h
xmlParser.cc
xmlParser.h

Příloha C

Konfigrační soubor

V této příloze, jsou připojeny konfigurační soubory pro třetí testovací topologii.

C.1 Soubor - omnetpp.ini

```
[General]
network = DIP_example
total-stack = 7MiB
tkenv-plugin-path = ../../../etc/plugins
sim-time-limit = 1day
debug-on-errors = false

# Routers IDs
**.R1.deviceId = "10.0.0.1"
**.R2.deviceId = "10.0.0.2"
**.R3.deviceId = "10.0.0.3"
**.R4.deviceId = "10.0.0.4"
**.R5.deviceId = "10.0.0.5"

**.LAN1.deviceId = "2014:0:5::10"
**.LAN2.deviceId = "2011:0:5::10"
**.LAN3.deviceId = "2012:0:5::10"
```

C.2 Soubor - config.xml

```
<Devices>
  <Router id="10.0.0.1">
    <Interfaces>
        <Interface name="eth0">
      <IPAddress>10.0.1.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:1::1/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
        </Interface>
        <Interface name="eth1">
      <IPAddress>10.0.11.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2015:0:5::1/56</IPv6Address>
```

```
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
        </Interface>
    </Interfaces>

    <Routing6>
      <OSPFv3 ProcessID="1">
        <RouterID>
          <IPAddress>10.0.0.1</IPAddress>
        </RouterID>
        <Areas>
          <Area AreaID="0">
            <Interfaces>
              <Interface name="eth0">
                <Type>Broadcast</Type>
                </Interface>
                <Interface name="eth1">
                  <Passive>Passive</Passive>
                </Interface>
            </Interfaces>
          </Area>
        </Areas>
      </OSPFv3>
    </Routing6>
</Router>

<Router id="10.0.0.2">
    <Interfaces>
      <Interface name="eth0">
        <IPAddress>10.0.1.2</IPAddress>
        <Mask>255.255.255.0</Mask>
        <IPv6Address>2001:0:0:1::2/64</IPv6Address>
        <IPv6Address>FF02::5/128</IPv6Address>
        <IPv6Address>FF02::6/128</IPv6Address>
      </Interface>
      <Interface name="ppp0">
        <IPAddress>10.0.2.1</IPAddress>
        <Mask>255.255.255.0</Mask>
        <IPv6Address>2001:0:0:2::1/64</IPv6Address>
        <IPv6Address>FF02::5/128</IPv6Address>
        <IPv6Address>FF02::6/128</IPv6Address>
      </Interface>
    </Interfaces>

    <Routing6>
      <OSPFv3 ProcessID="1">
        <RouterID>
          <IPAddress>10.0.0.2</IPAddress>
        </RouterID>
          <Areas>
            <Area AreaID="0">
            <Interfaces>
              <Interface name="eth0"></Interface>
              <Interface name="ppp0">
```

```
                <Type>PointToPoint</Type>
              </Interface>
            </Interfaces>
          </Area>
        </Areas>
      </OSPFv3>
    </Routing6>
</Router>

<Router id="10.0.0.3">
  <Interfaces>
    <Interface name="eth0">
      <IPAddress>10.0.1.3</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:1::3/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
    <Interface name="eth1">
      <IPAddress>10.0.33.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2011:0:5::1/56</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
      <Interface name="ppp0">
      <IPAddress>10.0.3.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:3::1/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
  </Interfaces>

  <Routing6>
    <OSPFv3 ProcessID="1">
      <RouterID>
         <IPAddress>10.0.0.3</IPAddress>
      </RouterID>
      <Areas>
        <Area AreaID="0">
        <Interfaces>
          <Interface name="eth0"></Interface>
            <Interface name="ppp0"></Interface>
            <Interface name="eth1">
              <Passive>True</Passive>
            </Interface>
          </Interfaces>
        </Area>
      </Areas>
    </OSPFv3>
  </Routing6>
</Router>
```

```
<Router id="10.0.0.4">
  <Interfaces>
    <Interface name="ppp0">
      <IPAddress>10.0.2.2</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:2::2/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
    <Interface name="ppp1">
      <IPAddress>10.0.3.2</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:3::2/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
    <Interface name="ppp2">
      <IPAddress>10.0.4.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2001:0:0:4::1/64</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
  </Interfaces>

  <Routing6>
    <OSPFv3 ProcessID="1">
      <RouterID>
        <IPAddress>10.0.0.4</IPAddress>
      </RouterID>
      <Areas>
        <Area AreaID="0">
          <Interfaces>
            <Interface name="ppp0"></Interface>
            <Interface name="ppp1"></Interface>
            <Interface name="ppp2"></Interface>
          </Interfaces>
        </Area>
      </Areas>
    </OSPFv3>
  </Routing6>
</Router>

<Router id="10.0.0.5">
  <Interfaces>
    <Interface name="eth0">
      <IPAddress>10.0.55.1</IPAddress>
      <Mask>255.255.255.0</Mask>
      <IPv6Address>2012:0:5::1/56</IPv6Address>
      <IPv6Address>FF02::5/128</IPv6Address>
      <IPv6Address>FF02::6/128</IPv6Address>
    </Interface>
    <Interface name="ppp0">
      <IPAddress>10.0.3.1</IPAddress>
```

```
        <Mask>255.255.255.0</Mask>
        <IPv6Address>2001:0:0:4::2/64</IPv6Address>
        <IPv6Address>FF02::5/128</IPv6Address>
        <IPv6Address>FF02::6/128</IPv6Address>
      </Interface>
    </Interfaces>

    <Routing6>
      <OSPFv3 ProcessID="1">
        <RouterID>
          <IPAddress>10.0.0.5</IPAddress>
        </RouterID>
        <Areas>
          <Area AreaID="0">
            <Interfaces>
              <Interface name="ppp0"></Interface>
              <Interface name="eth0">
                <Passive>True</Passive>
              </Interface>
            </Interfaces>
          </Area>
        </Areas>
      </OSPFv3>
    </Routing6>
  </Router>

  <Host id="2014:0:5::10">
    <Interfaces>
      <Interface name="eth0">
        <IPv6Address>2014:0:5::10/56</IPv6Address>
      </Interface>
    </Interfaces>
    <DefaultRouter6>2014:0:5::1</DefaultRouter6>
  </Host>

  <Host id="2012:0:5::10">
    <Interfaces>
      <Interface name="eth0">
        <IPv6Address>2012:0:5::10/56</IPv6Address>
      </Interface>
    </Interfaces>
    <DefaultRouter6>2012:0:5::1</DefaultRouter6>
  </Host>

  <Host id="2011:0:5::10">
    <Interfaces>
      <Interface name="eth0">
        <IPv6Address>2011:0:5::10/56</IPv6Address>
      </Interface>
    </Interfaces>
    <DefaultRouter6>2011:0:5::1</DefaultRouter6>
  </Host>
</Devices>
```