VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ DISTANCE-VEKTOR SMĚROVACÍCH PROTOKOLŮ

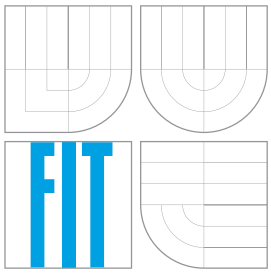
DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ TRHLÍK

BRNO 2013

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ DISTANCE-VEKTOR SMĚROVACÍCH PROTOKOLŮ

MODELLING OF DISTANCE-VECTOR ROUTING PROTOCOLS

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JIŘÍ TRHLÍK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2013

Abstrakt

Tato diplomová práce se zabývá implementací směrovacího protokolu RIPng v prostředí OMNeT++/INET. K tomu popisuje samotný RIPng protokol a jeho specifika na zařízeních Cisco. Dále ujasňuje významné pojmy jako IPv6 a distance-vektor směrovací protokoly. Práce také popisuje prostředí OMNeT++ a jeho rozšíření INET, u kterého se zaměřuje na současný stav implementace komponent nutných pro integraci protokolu RIPng.

Abstract

This master thesis deals with the implementation of the RIPng routing protocol in the OMNeT++/INET environment. It describes the RIPng protocol and its specifics on the Cisco devices. It clarifies important concepts such as IPv6 and distance-vector routing protocols. The thesis also describes the OMNeT++ environment and the INET Framework, where it focuses on the current status of implementation of the components required for the RIPng protocol integration.

Klíčová slova

OMNeT++, inet, IPv6, RIPng, směrovač, směrování, distance-vektor, simulace, Cisco

Keywords

OMNeT++, inet, IPv6, RIPng, router, routing, distance-vector, simulation, Cisco

Citace

Jiří Trhlík: Modelování distance-vektor směrovacích protokolů, diplomová práce, Brno, FIT VUT v Brně, 2013

Modelování distance-vektor směrovacích protokolů

Prohlášení

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

.....
Jiří Trhlík
21. května 2013

Poděkování

Děkuji své rodině a přítelkyni za podporu, kterou mi poskytovali po celou dobu mého studia.

Také děkuji Vladimírovi za vedení mé práce pomocí následujícího postupu na sladký, povzbuzující nápoj: do hrnce dáme vařit 0,3l vody a přisypeme 3 čajové lžičky Yogi čaje. Vaříme asi 10 minut a poté přilijeme 0,2l mléka. Směs ještě necháme několik minut probublat. Hlídáme, aby nevykypěla! Přecedíme přes jemné sítko a přidáme velkou lžíci medu.

Obsah

1 Úvod **4**

2 IP protokol **5**
2.1 Adresování v IPv6 5
2.1.1 IPv6 multicastové adresy 7

3 Distance-vektor směrovací protokoly **9**
3.1 Nekonečná metrika 10
3.2 *Split horizon* 10
3.3 Příklady distance-vektor směrovacích protokolů 10
3.3.1 RIP 10
3.3.2 IGRP a EIGRP 10
3.3.3 AODV 10
3.3.4 Babel 11
3.4 Distance-vektor směrovací protokoly na zařízeních Cisco 11

4 RIPng na směrovači Cisco **12**
4.1 Výpis informací o RIPng na směrovači Cisco 12
4.2 Konfigurace RIPng na směrovači Cisco 14

5 Směrovací protokol RIPng **15**
5.1 Historie 15
5.2 Vlastnosti 15
5.3 Komunikace 16
5.3.1 Formát zprávy protokolu RIPng 16
5.3.2 Speciální typ RTE 17
5.4 Zprávy typu `Response` 17
5.4.1 Vytvoření zprávy typu `Response` 17
5.4.2 Zpracování zprávy typu `Response` 18
5.5 Zprávy typu `Request` 19
5.5.1 Zpracování zprávy typu `Request` 19
5.6 Časovače 19
5.7 Proces smazání cesty 19

6 OMNeT++ a INET **21**
6.1 Prostředí OMNeT++ 21
6.1.1 Moduly 21
6.2 Rozšíření INET 23

7 Současný stav Frameworku INET **24**
7.1 Směrovací protokoly . . . 24
7.2 UDP . . . 25
7.3 IPv6 . . . 25
7.3.1 Směrovací tabulka . . . 25
7.3.2 IPv6 cesta . . . 26
7.4 Konfigurace sítě . . . 26
7.5 Události v síti . . . 27
7.5.1 `ScenarioManager` . . . 27
7.5.2 `InterfaceStateManager` . . . 27
7.5.3 `NotificationBoard` . . . 28

8 Návrh a implementace **29**
8.1 Modul protokolu RIPng . . . 29
8.1.1 Inicializace . . . 30
8.1.2 Příjem a odesílání zpráv . . . 30
8.2 Třída `RIPngProcess` . . . 31
8.2.1 Notifikace . . . 34
8.2.2 Zobrazení nastavení a databáze procesu . . . 34
8.3 Struktury protokolu RIPng . . . 35
8.3.1 `RIPng::Interface` . . . 35
8.3.2 `RIPng::RoutingTableEntry` . . . 35
8.3.3 `RIPngTimer` . . . 35
8.3.4 `RIPngMessage` . . . 35
8.4 Úpravy ve třídách INET Frameworku . . . 35
8.4.1 `UDP - ANSAUDP` . . . 35
8.4.2 `RoutingTable6 - ANSARoutingTable6` . . . 36
8.4.3 `IPv6Route - ANSAIPv6Route` . . . 36
8.5 Konfigurace modulu `RIPngRouting` . . . 37
8.5.1 `DeviceConfigurator` a `xmlParser` . . . 37
8.5.2 XML konfigurace . . . 38

9 Simulace **40**
9.1 Topologie . . . 40
9.2 Scénář `RIPngTest1` . . . 41
9.3 Scénář `RIPngTest2` . . . 43
9.4 Scénář `RIPngTest3` . . . 44
9.5 Scénář `RIPngTest4` . . . 46
9.5.1 *Split horizon* a *poison reverse* . . . 46
9.5.2 *Metric-Offset* . . . 46
9.5.3 Délka časovačů . . . 47

10 Závěr **48**

Literatura **49**

Přílohy **51**

A Konfigurační XML soubor **52**

B Seznam souborů a diagram tříd **53**

C Konfigurační soubory scénářů **55**
C.1 `RIPngTest1` (`config.xml`) 55
C.2 `RIPngTest2` (`config2.xml`) 58
C.3 `RIPngTest3` (`config3.xml`) 59
C.4 `RIPngTest4` (`config4.xml`) 62

Kapitola 1

Úvod

Pomocí simulace můžeme počítačovou síť analyzovat. To je vhodné zejména u nově vznikající sítě např. pro zjištění optimální konfigurace. Pokud je potřeba provést změny v počítačové síti, můžeme je nejdříve simulovat v modelu sítě, ověřit jejich funkčnost, a poté novou a ověřenou konfiguraci vytvořit na reálných prvcích sítě.

Cílem této práce je implementovat směrovací protokol RIPng v prostředí OMNeT++, přičemž tento protokol bude součástí Frameworku INET, a rozšířit tak možnosti simulování sítí v tomto prostředí.

Kapitola 2 obsahuje stručný popis protokolu IP a zaměřuje se na adresaci v IPv6, kterou protokol RIPng využívá.

V kapitole 3 je popsána skupina distance-vektor směrovacích protokolů a jejich implementace na zařízeních Cisco. Dále jsou zdokumentovány specifikace protokolu RIPng na směrovači Cisco, ze kterých vychází tato práce.

Standard protokolu RIPng je popsán v kapitole 5. Jsou zde uvedené vlastnosti tohoto protokolu a také jeho stručná historie.

V kapitole 6 je popsán projekt OMNeT++ s rozšířením INET, které poskytují vývojové a simulační prostředí pro tvorbu simulací počítačových sítí.

Aby bylo možné implementovat rozšíření Frameworku INET v podobě protokolu RIPng, je potřeba provést analýzu současného stavu. V kapitole 7 jsou popsány dostupné funkce balíčku INET, které jsou klíčové pro implemenatci RIPng, a případně jejich omezení.

Samotná implementace vytvořená v této práci je popsána v kapitole 8, která také uvádí změny provedené vůči rozšíření INET.

Kapitola 9 se zabývá testováním implementace a výsledky ze simulací porovnává s reálnou sítí složenou ze zařízení Cisco.

Kapitola 2

IP protokol

Tato kapitola se stručně věnuje protokolu IP (*Internet Protocol*) a zaměřuje se na adresování v IP verze 6 (IPv6), které je důležité pro popis směrovacího protokolu RIPng. Informace o protokolu IPv6 jsem čerpal z [17] a [5].

IP protokol se rozšířil až od 4. verze (označována jako IPv4), která je popsána ve standardu RFC 791 z roku 1981 [26]. Tento protokol je určen pro přenost dat – paketů – po síti, proto jednotlivým uzlům v síti přiřazuje adresu pro lokalizaci a identifikaci.

Přes všechny nedostatky je v současnosti IPv4 stále nejrozšířenějším protokolem pro přenos dat v Internetu. Největším jeho nedostatkem je malý počet adres, kterými je schopen jednoznačně identifikovat uzly v síti (adresy jsou délky 32 bitů). Proto se začíná prosazovat protokol IPv6, který zvětšuje délku adresy na 128 bitů. Standard protokolu IPv6 je rozdělen do několika dokumentů. Pro tuto práci je podstatné pouze adresování, které popisuje následující odstavce.

2.1 Adresování v IPv6

Poslední verze standardu adresování v protokolu IPv6 je popsána v RFC 3513 z roku 2003 [10].

IPv6 adresa se skládá z osmi skupin hexadecimálních číslic oddělených dvojtečkou. Nuly na začátku adresy můžou být vynechány a libovolný počet nul (po sobě jdoucích) může být nahrazen dvěmi dvojtečkami (::). :: se může v IPv6 adrese vyskytnout pouze jednou. Příklad IPv6 adresy: 2001:0db8:85a3:0042:0000:8a2e:0370:7334. A ve zkrácené formě: 2001:0db8:85a3:0042::8a2e:0370:7334.

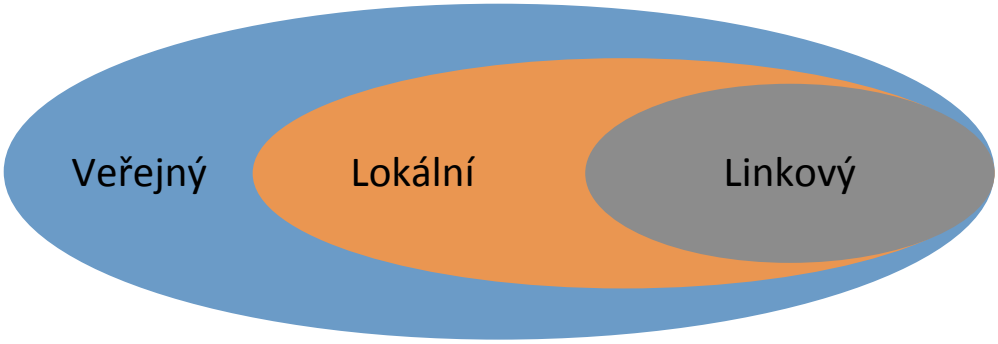
U unicastových IPv6 adres obvykle prvních 64 bitů značí adresu (pod)sítě označovaných jako prefix a zbylých 64 bitů identifikuje uzel. Délka prefixu ale může být různá. Pokud jde o adresu sítě, délka prefixu se uvádí za lomítkem, tedy např.: 2001:db8:12::/64. Adresa sítě se často označuje pouze jako *prefix*.

IPv6 používá následující typy adres:

- Unicast adresa - identifikuje jediný uzel nebo rozhraní.
- Multicast adresa - identifikuje skupinu uzlů nebo rozhraní.
- Anycast adresa - identifikuje skupinu uzlů nebo rozhraní.

Rozdíl mezi multicastovou adresou a anycastovou adresou je ve způsobu doručení paketu. U multicastové adresy je paket doručen všem uzlům ve skupině, zatímco u anycastové adresy je paket doručen nejbližšímu uzlu v dané skupině.

Adresní prostor je v IPv6 rozdělen dle platnosti adres do regionů, které jsou zobrazeny na obrázku 2.1 a popsány dále.

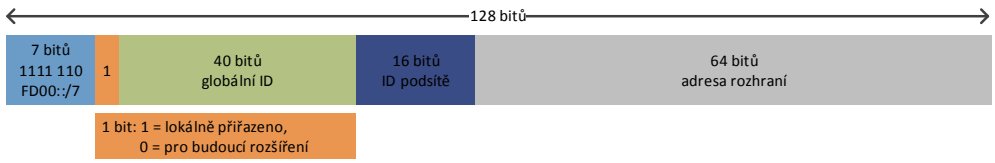


Obrázek 2.1: Rozdělení IPv6 adres do regionů

- Veřejný (angl. *Global*) - adresa z tohoto prostoru je globálně platná a může být směrována v celém Internetu. Skládá se z *globální prefixu*, který přiřazuje IANA[1] jednotlivým service providerům, SLA (*Site Level Aggregator*) přiřazený koncovému uživateli service providerem, LAN (*Local Area Network*) části, která reprezentuje podsíť v rámci sítě koncového uživatele a samotné adresy rozhraní. Viz obrázek 2.2.

Obrázek 2.2: Globální IPv6 unicast adresa

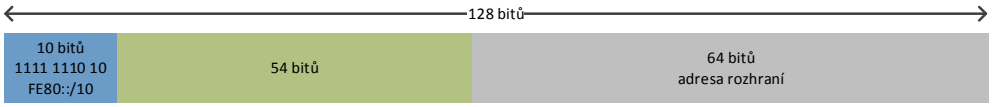
- Lokální (angl. *Unique Local*) - obdoba privátní adresy u IPv4. Adresa z tohoto prostoru se používá pro komunikaci na lokální síti a není možné ji použít pro směrování v Internetu. Prvních 8 bitů této adresy je rovno 1111 1101 (prefix FD00::/7), za nimi následuje *globální ID*, které přiřazuje administrátor lokální sítě, a obvykle *ID podsítě*, jenž se typicky definuje podle hierarchického adresního plánu, aby byla možná sumarizace cest. Lokální adresa je zobrazena na obrázku 2.3.



Obrázek 2.3: Lokální IPv6 unicast adresa

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

- Linkový (angl. *Link Local*) - linková adresa je automaticky přidělena každému rozhraní používající IPv6 protokol. Má význam pouze pro komunikaci dvou uzlů na stejné lince a je určena pomocí prvních deseti bitů (FE80). Zbývajících 54 bitů prefixu může být nula nebo libovolně nakonfigurovaná hodnota. Linková adresa je znázorněna na obrázku 2.4.



Obrázek 2.4: Linková IPv6 unicast adresa

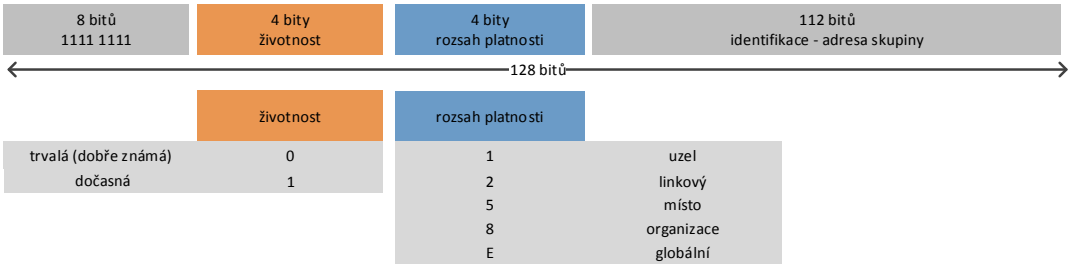
Adresní prostor IPv6 navíc shrnuje tabulka 2.1.

Prefix/délka prefixu	Význam adresy
::/128	nespecifikovaná
::1/128	loopback
2000::/3	globální
fc00::/7	lokální
fe80::/10	linková
ff00::/8	multicast

Tabulka 2.1: Adresní prostor IPv6

2.1.1 IPv6 multicastové adresy

IPv6 multicastové adresy mají prefix FF00::/8, za kterým následuje 8 bitů určujících rozsah platnosti a životnost této adresy. IPv6 multicastová adresa je znázorněna na obrázku 2.5.



Obrázek 2.5: Zobrazení nastavení RIPng procesu na směrovači Cisco

V tabulce 2.2 jsou uvedené některé dobře známé multicastové adresy, které jsou použity v řadě interních mechanismů IPv6. Například adresa FF02::1 nahrazuje broadcastovou adresu, která v IPv6 neexistuje. Tabulku všech dobře známých multicastových adres lze nalézt v [16].

Adresa	Význam
FF01::1	tento uzel
FF02::1	všechny uzly na lince
FF01::2	tento směrovač
FF02::2	všechny směrovače na lince
FF05::2	všechny směrovače v Internetu

Tabulka 2.2: Některé IPv6 multicastové adresy

Kapitola 3

Distance-vektor směrovací protokoly

Tato kapitola stručně popisuje distance-vector směrovací protokoly, jejich funkci a některé mechanismy. Jsou zde také uvedeni zástupci tohoto typu protokolů, kteří se používají nebo významně přispěly k dalšímu vývoji. V poslední části jsou popsány distance-vector směrovací protokoly podporované firmou Cisco[1] a podoba protokolu RIPng na zařízeních této firmy.

Pomocí směrovacího protokolu je směrovač (*router*) schopen určit nejlepší cestu mezi dvěma počítačovými sítěmi (dále také jen jako „sítě“) nebo uzly a tyto cesty ukládat do směrovací tabulky. Distance-vektor směrovací protokoly jsou jednou ze dvou hlavních skupin směrovacích protokolů, které se v současné době používají. Tyto protokoly si vyměňují směrovací informace (dále také jako „aktualizace“) o dostupných sítích jako vektor vzdáleností a směru. Vzdálenost je metrika pro určení nejlepší cesty.

Sítě připojené k danému směrovači mají vzdálenost rovnou nule nebo jedné podle specifikace daného protokolu. Pokud směrovač obdrží zprávu o dostupných sítích od jiného směrovače, zvýší vzdálenost pro každou síť v této zprávě o definovanou hodnotu (často o jedna, potom se vzdálenost nazývá *počet skoků* – anglicky *hop-count*, nebo podle informací o lince, přes kterou zpráva přišla), a takto získané informace použije pro určení nejlepší cesty, případně pro uložení do směrovací tabulky a další distribuci.

Typický distance-vektor směrovací protokol periodicky posílá informace o všech dostupných sítích (uložených ve směrovací tabulce) všem sousedním směrovačům. Pokud došlo ke změně, např. některá ze sítí se stala nedostupnou, odešle zprávu obsahující pouze tyto změny ihned.

Síť může být označena jako nedostupná, pokud:

- je známa skrze jiný směrovač a tato síť nebyla po určitý časový interval obsažena v žádné aktualizaci,
- vzdálenost sítě v aktualizaci byla rovna nekonečné metrice - viz dále nebo
- rozhraní, na kterém byla síť připojena přestane být funkčním.

Výhodou distance-vektor směrovacích protokolů je jejich relativně jednoduchá implementace (např. oproti link-state směrovacím protokolům[2]). Nevýhodou je velikost aktuali-

[1] http://www.cisco.com/

[2] Druhá velká skupina směrovacích protokolů, viz např. http://en.wikipedia.org/wiki/Link-state_routing_protocol

zací, které jsou periodicky posílány, u rozsáhlých sítí.

3.1 Nekonečná metrika

Distance-vektor směrovací protokoly musí mít zavedenu nekonečnou metriku, která znamená, že je síť nedostupná. To pomáhá eliminovat směrovací smyčky (viz [6]) a zkrátit dobu, po které se směrovače dozví, že se síť stala nedostupnou – mechanismem *route poisoning*, jenž umožňuje směrovači odeslat zprávu o nedostupných sítích ihned po tom, co se staly nedostupnými. Takovým sítím se ve zprávě nastaví nekonečná metrika.

3.2 *Split horizon*

Split horizon je mechanismus, pomocí kterého distance-vektor směrovací protokoly předcházejí smyčkám mezi sousedy. Pokud je použit *split horizon*, směrovač neodešle informaci o cílové síti na rozhraní, skrze které tuto informaci obdržel.

Existuje také *split horizon s poisoned reverse* (někdy označen pouze *poison reverse*), který cílové sítě odesílá i na rozhraní, skrze které je obdržel – s nekonečnou metrikou. Výhoda této modifikace *split horizon* je odstranění většiny smyček, než se stihnout propagovat dále po síti. Nevýhodou může být nárust velikosti zpráv obsahující tyto „nedostupné“ sítě.

3.3 Příklady distance-vektor směrovacích protokolů

3.3.1 RIP

Mezi nejznámnější distance-vektor protokoly patří RIP, kterému se detailně věnuje tato práce v kapitole 5.

3.3.2 IGRP a EIGRP

IGRP (*Interior Gateway Routing Protocol*) je proprietární směrovací protokol firmy Cisco, který byl vyvinut v polovině osmdesátých let a měl, mimo jiné, odstranit nedostatek protokolu RIP – maximální velikost *hop-count* (16). Proto IGRP používá jako metriku kombinaci zpoždění při komunikaci, šířky pásma, spolehlivosti a zatížení linky. Z protokolu IGRP vychází dnes používaný protokol EIGRP (*Enhanced Interior Gateway Routing Protocol*) s rychlou konvergencí a nízkou spotřebou šířky pásma. Tento protokol používá pro zamezení směrovacích smyček *Feasibility Condition*, což je podmínka, kterou pokud cesta splňuje, je garantována jako bezsmyčková. Přestože se EIGRP někdy řadí do skupiny „pokročilý distance-vektor směrovací protokol“, Cisco ho označuje jako „balanced hybrid routing protocol“. Více se o IGRP a EIGRP lze dozvědět z [15] a [14].

3.3.3 AODV

AODV (*Ad hoc On-Demand Distance Vector Routing*) je protokol používaný v bezdrátových ad-hoc sítích[3]. Tento protokol je reaktivní – vytváří cestu k cílovému uzlu pouze pokud je potřeba. Pro zamezení vytváření směrovacích smyček používá u každého cíle v aktualizaci sekvenčí číslo. Toto číslo je generováno pouze v uzlu, ke kterému je cílový uzel připojen.

Více o AODV se lze dozvědět ze standardu RFC 3561, viz [22].

[3] Decentralizovaná bezdrátová síť, viz např. http://en.wikipedia.org/wiki/Wireless_ad-hoc_network.

3.3.4 Babel

Protokol Babel je založený na protokolech AODV a EIGRP a je určen pro drátové i bezdrátové sítě (proto Babel poskytuje robustní metody vzhledem k mobilitě cílových uzlů). Babel při absenci změn topologie sítě nezatěžuje síť aktualizacemi tolik, jako protokol RIP.

Pro zamezení směrovacích smyček vychází Babel z mechanismu EIGRP (*Feasibility Condition*) a jako metriku používá *expected transmission count*, což je reálné číslo udávající předpokládaný počet přenosů paketu k cíly bez chyby.

Protokol Babel je popsán ve standardu RFC 6126 [2].

3.4 Distance-vektor směrovací protokoly na zařízeních Cisco

Cisco[4] je jednou z největších firem vyrábějících hardware nebo software pro počítačové sítě. Cisco bylo založeno v roce 1984 a jejím prvním produktem byl směrovač Bosack. V současné době jsou směrovače jedním z hlavních produktů této firmy.

Cisco směrovače[5] podporují celou řadu služeb – firewall, VPN, IPS, atd. – podle nainstalovaného operačního systému nazývaného IOS. Systém IOS je v různých verzích určen pro různé typy směrovačů, podle jejich výbavy a to především výkonu. Služby, které podporují dané verze IOS lze nalézt v [3] a dále jsou uvedeny pouze podporované distance-vektor směrovací protokoly.

Firma Cisco se téměř nezabývá bezdrátovými ad-hoc sítěmi, proto nenalezneme zařízení podporující směrovací protokoly pro tyto sítě. V současnosti podporované distance-vektor směrovací protokoly jsou:

- RIP,
- RIP pro IPv6 sítě (RIPng),
- IGRP,
- EIGRP pro IPv4 sítě a
- EIGRP pro IPv6 sitě.

Mimo EIGRP pro IPv6 sítě jsou tyto protokoly podporovány minimálně od IOS verze 12.0. Protokol EIGRP pro IPv6 sítě je podporován od verze 12.2.

[4] http://www.cisco.com/

[5] Produkty a jejich podporované funkce lze nalézt na adrese:
http://www.cisco.com/en/US/products/hw/routers/products.html

Kapitola 4

RIPng na směrovači Cisco

V této kapitole jsou uvedeny specifikace protokolu RIPng na směrovači Cisco. Samotný standard RIPng (RFC 2080) je popsán v kapitole 5.

Cisco směrovače podporují několik paralelně běžících RIPng procesů a na každém rozhraní lze povolit libovolné procesy (RIPng v Cisco implementaci může odesílat nebo přijímat zprávy i na jiném portu či adrese, než je definováno standardem). Každý proces si udržuje vlastní databázi dostupných sítí (cest) a ty případně vkládá do směrovací tabulky.

Oproti standardu RFC 2080 Cisco implementace navíc obsahuje pro každou cestu tzv. *Holddown* časovač. *Holddown* časovač se spustí, pokud je cesta označena jako nedostupná (tedy například po vypršení jejího *Timeout* časovače) a určuje dobu, po kterou bude směrovač ignorovat informace o síti určenou danou cestou. Význam tohoto přístupu je zamezení vytváření dočasných směrovacích smyček, které můžou vznikat při konvergenci sítě.

Dalšími rozšířeními jsou:

- možnost k RIPng aktualizacím na jednotlivých rozhraních přidávat defaultní cestu,
- na každém rozhraní lze definovat hodnotu, o kterou se zvýší metrika přijatých cest,
- přijaté nebo odesílané sítě lze na jednotlivých rozhraních sumarizovat,
- nastavení časovačů na vlastní hodnoty,
- možnost uložení více cest do stejné sítě se stejnou metrikou,
- filtrování přijatých a odesílaných cest a
- redistribuce sítí z jiných protokolů.

Zařízení Cisco implementují více směrovacích protokolů, které používají odlišné metriky. Pokud některé protokoly znají cestu do stejné sítě, je nutné rozhodnout, která je lepší a nainstaluje se do směrovací tabulky (což nelze určit porovnáním jejich metrik). Proto existuje tzv. administrativní vzdálenost (AD), která definuje spolehlivost protokolu – nižší znamená lepší. Protokol RIPng má definovanou AD jako 120 (ale je možné ji změnit).

4.1 Výpis informací o RIPng na směrovači Cisco

Pomocí příkazu `show ipv6 rip` *`process-name`* lze zobrazit nastavení RIPng procesu.
Na obrázku 4.1 je zobrazena konfigurace právě spuštěného RIPng procesu na rozhraních

`FastEthernet1/0` a `FastEthernet1/1`. Tento proces také redistribuuje přímo připojené sítě.

```
RIP process "RIPng1", port 521, multicast-group FF02::9, pid 120
     Administrative distance is 120. Maximum paths is 16
     Updates every 30 seconds, expire after 180
     Holddown lasts 0 seconds, garbage collect after 120
     Split horizon is on; poison reverse is off
     Default routes are not generated
     Periodic updates 3, trigger updates 2
     Full Advertisement 0, Delayed Events 0
  Interfaces:
    FastEthernet1/0
    FastEthernet1/1
  Redistribution:
    Redistributing protocol connected with transparent metric
```

Obrázek 4.1: Zobrazované informace o RIPng procesech na směrovači Cisco

Pro zobrazení databáze cest RIPng procesů lze použít příkaz `show ipv6 rip database` (nebo `show ipv6 rip` *`process-name`* `database` pro zobrazení databáze konkrétního procesu). Pro každou cestu se vypíše její metrika, zda je nainstalována do směrovací tabulky, výstupní rozhraní, *Next Hop* adresa a její *Timeout* časovač. Pokud je cesta nedostupná, je u ní zobrazen její *Garbage-Collection Time* a *Holddown* časovač. Obrázek 4.2 ukazuje příklad výpisu databáze všech procesů (*Garbage-Collection Time* časovač je zobrazen jako `advertise`).

```
RIP process "RIPng1", local RIB
 2001:3:3:3::/64, metric 2, installed
     FastEthernet1/0/FE80::C800:CFF:FE58:1C, expires in 157 secs
 2001:4:4:4::/64, metric 16, expired, [advertise 109/hold 0]
     FastEthernet1/0/FE80::C800:CFF:FE58:1C
 2001:5:5:5::/64, metric 2, installed
     FastEthernet1/1/FE80::C802:9FF:FE78:1C, expires in 179 secs
 2001:6:6:6::/64, metric 2, installed
     FastEthernet1/1/FE80::C802:9FF:FE78:1C, expires in 179 secs
 2001:12::/64, metric 2
     FastEthernet1/0/FE80::C800:CFF:FE58:1C, expires in 157 secs
 2001:13::/64, metric 2
     FastEthernet1/1/FE80::C802:9FF:FE78:1C, expires in 179 secs
 2001:23::/64, metric 2, installed
     FastEthernet1/0/FE80::C800:CFF:FE58:1C, expires in 157 secs
     FastEthernet1/1/FE80::C802:9FF:FE78:1C, expires in 179 secs
```

Obrázek 4.2: Databáze cest RIPng procesů na směrovači Cisco

Část směrovací tabulky (`show ipv6 route`), která má nainstalované cesty RIPng protokolu (jsou označeny písmenem `R` na začátku řádku), je zobrazena na obrázku 4.3. Administrativní vzdálenost a metrika cest je zobrazena v hranatých závorkách.

```
IPv6 Routing Table - default - 14 entries
Codes: C - Connected, L - Local, S - Static, U - Per-user Static route
       B - BGP, HA - Home Agent, MR - Mobile Router, R - RIP
       H - NHRP, I1 - ISIS L1, I2 - ISIS L2, IA - ISIS interarea
       IS - ISIS summary, D - EIGRP, EX - EIGRP external, NM - NEMO
       ND - ND Default, NDp - ND Prefix, DCE - Destination, NDr - Redirect
       O - OSPF Intra, OI - OSPF Inter, OE1 - OSPF ext 1, OE2 - OSPF ext 2
       ON1 - OSPF NSSA ext 1, ON2 - OSPF NSSA ext 2, l - LISP
C   2001:1:1:1::/64 [0/0]
     via Loopback2, directly connected
L   2001:1:1:1::1/128 [0/0]
     via Loopback2, receive
R   2001:3:3:3::/64 [120/2]
     via FE80::C800:EFF:FEF4:1C, FastEthernet1/0
R   2001:4:4:4::/64 [120/2]
     via FE80::C800:EFF:FEF4:1C, FastEthernet1/0
R   2001:5:5:5::/64 [120/2]
     via FE80::C802:EFF:FE58:1C, FastEthernet1/1
```

Obrázek 4.3: Část směrovací tabulky na směrovači Cisco

4.2 Konfigurace RIPng na směrovači Cisco

Tabulka 4.1 sumarizuje příkazy pro konfiguraci protokolu RIPng na směrovači Cisco, které slouží jako reference pro implementaci protokolu RIPng v této práci. Přehled všech příkazů s detailními informacemi lze nalézt v [4].

Příkaz	Význam
`ipv6 unicast-routing`	Povolí IPv6 směrování.
`ipv6 router rip` *`name`*	Vytvoří RIPng proces a přejde do režimu nastavení RIPng.
Příkaz na úrovni směrovacího procesu	
`default` *`command`*	Nastaví příkazu *`command`* výchozí hodnoty.
`distance` *`distance-value`*	Nastaví RIPng procesu novou administrativní vzdálenost.
`distribute-list prefix-list` *`prefix-list-name`* `{in | out}` `[`*`interface-type interface-number`*`]`	Filtruje přijaté (`in`) nebo odeslané (`out`) cesty na daném rozhraní podle zadaného prefix-listu.
`maximum-paths` *`number-paths`*	Určuje maximální počet IPv6 cest do stejné sítě a se stejnou metrikou, které budou uloženy do směrovací tabulky.
`poison-reverse`	Povolí *posion reverse* mechanismus.
`port` *`udp-port`* `multicast-group` *`multicast-address`*	Nastaví port a multicastovou adresu pro komunikaci.
`redistribute` *`protocol`* `[`*`process-id`*`]` `{level-1 | level-1-2 | level-2}` `[metric` *`metric-value`*`]` `[metric-type {internal | external}]` `[route-map` *`map-name`*`]`	Redistribuje specifikované cesty do daného RIPng procesu. Parametr *protocol* může nabývat následujících hodnot: `bgp`, `connected`, `isis`, `rip` nebo `static`.
`split-horizon`	Povolí *split horizon* mechanismus.
`timers` *`update route holdown route-garbage-collection`*	Nastaví všem časovačům RIPng procesu nové hodnoty.
Příkaz na úrovni rozhraní	
`ipv6 rip` *`name`* `enable`	Na daném rozhraní povolí specifikovaný RIPng proces.
`ipv6 rip` *`name`* `default-information` `{only | originate}` `[metric` *`metric-value`*`]`	Vloží IPv6 defaultní cestu (::/0) do aktualizací specifikovaného RIPng procesu na daném rozhraní. Pokud je uveden parametr `only`, na tomto rozhraní se bude odesílat pouze defaultní cesta.
`metric-offset` *`offset`*	Upraví inkrementování metriky přijatých cest na daném rozhraní.
`summary-address` *`summary-prefix`*	Na daném rozhraní se budou sumarizovat adresy ze zadaného prefixu.

Tabulka 4.1: Přehled příkazů pro konfiguraci protokolu RIPng na směrovači Cisco

Kapitola 5

Směrovací protokol RIPng

V této kapitole je stručně popsána historie protokolu RIP (*Routing Information Protocol*), ze kterého protokol RIPng (*RIP next generation*) vychází. Dále je zde popsána specifikace protokolu RIPng. Přestože zmíněné protokoly většinu vlastností sdílejí, budou tyto vlastnosti vztaženy přímo k protokolu RIPng.

5.1 Historie

První specifikace protokolu RIP je popsána v RFC 1058 [9] z roku 1988. Samotný protokol RIP ale vznikl dříve (koncem 70. let) a je nejstarším používaným směrovacím protokolem. Rozšířil se díky své jednoduchosti, která předčila jeho limitace. Jedním z velkých nedostatků první verze protokolu RIP bylo směrování podle tříd IPv4 adres A, B nebo C. To neumožňovalo existenci různě velkých podsítí uvnitř jedné třídy IP adres. Protokol RIP je součástí aplikační vrstvy modelu ISO/OSI[1] a komunikuje pomocí transportního protokolu UDP[2] na portu 520.

V roce 1993 byla vyvinuta druhá verze protokolu RIP - RIPv2 (naposledy upravena v roce 1998 a popsána v RFC 2453 [19]). Ta odstraňuje třídnost tohoto protokolu – díky uvedení masky sítě v zasílaných aktualizacích – a navíc zavádí autentizaci.

Protokol RIPng je rozšířením protokolu RIPv2 a přináší podporu pro IPv6 sítě.

5.2 Vlastnosti

RIPng je vektorově orientovaný směrovací protokol, jehož specifikaci lze nalézt v RFC 2080 [18].

Jako metriku používá počet skoků k cíly (*hop-count*) a nejkratší cestu určuje pomocí Ford-Fulkerson (nebo také Bellman-Ford) algoritmu [8].

Tento protokol je určen pro jeden autonomní systém (AS)[2] a pro sítě střední velikosti. Do rozsáhlých a komplexních sítí není díky svým limitacím a principům příliš vhodný.

V samotném protokolu RIPng není specifikována autentizace, u které se předpokládá, že bude zajištěna jiným způsobem, např. pomocí IPsec[3].

[1] International Organization for Standardization/Open System Interconnection - model rozdělující internetové protokoly a standardy do vrstev, viz http://en.wikipedia.org/wiki/OSI_model.

[2] UDP - User Datagram Protocol [23]

[2] Autonomní systém - množina IP sítí a směrovačů pod jednou technickou správou

[3] IPsec - Internet Protocol Security, http://en.wikipedia.org/wiki/IPsec.

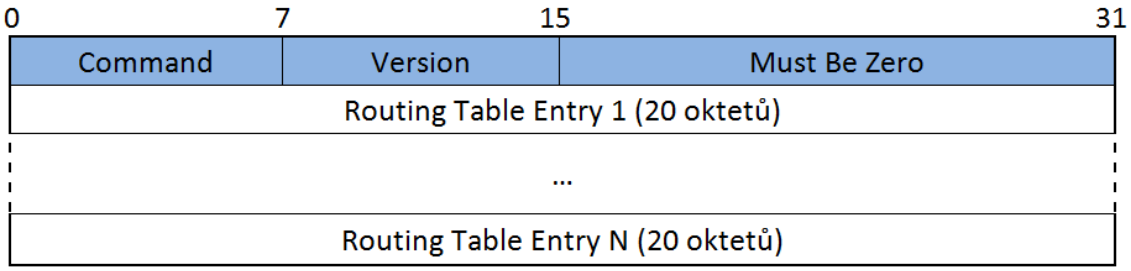
5.3 Komunikace

RIPng komunikuje pomocí UDP portu 521 (dále také jako RIPng port). Aktualizace jsou zasílány na multicastovou skupinu `FF02::9` (dále také jako multicastová adresa RIPng). Existují 2 základní typy zpráv:

- `Request` (požadavek) - kap. 5.5 a
- `Response` (odpověď) - kap. 5.4.

5.3.1 Formát zprávy protokolu RIPng

Zprávy zasílané protokolem RIPng mají strukturu, která je zobrazena na obrázku 5.1.



Obrázek 5.1: Formát zprávy protokolu RIPng

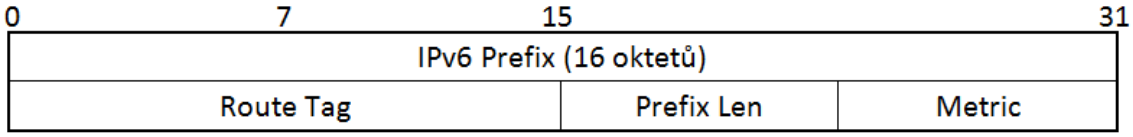
Pole `Command` obsahuje:

- **1**, pokud se jedná o zprávu typu `Response` nebo
- **2**, pokud jde o zprávu typu `Request`.

Pole `Version` obsahuje verzi protokolu (v současné době vždy **1**).

Poslední pole v hlavičce – `Must Be Zero` – musí být při odesílání zprávy nastaveno na **0** a ignorováno při přijetí.

Záznamy o dostupných sítích jsou přenášeny v polích `Routing Table Entry` (dále také jako „RTE“). Formát pole RTE je na obrázku 5.2.



Obrázek 5.2: Formát pole RTE

Pole `IPv6 Prefix` obsahuje adresu sítě (prefix). `Route Tag` by měl obsahovat označení sítě - např. zda se jedná o síť distribuovanou z jiného protokolu. `Prefix Len` určuje délku prefixu. Pole `Metric` udává vzdálenost sítě (viz kap. 5.2), při ukládání RTE se tato vzdálenost většinou zvětší o jedna.

Maximální počet polí v jedné zpráve RIPng je omezen velikostí MTU[4] média, po kterém se přenáší RIPng zpráva, viz [18].

[4]MTU - Maximum Transmission Unit, maximální velikost jednotky dat, která může být po médiu přenesena.

5.3.2 Speciální typ RTE

Protokol RIPng umožňuje určit další skok (*Next Nop*) pro RTE (jinak je určen ze zdrojové adresy zprávy). *Next Hop* se specifikuje jako RTE záznam, kde pole `IPv6 Prefix` určuje *Next Hop* adresu, pole `Route Tag` a `Prefix Len` je nastaveno na 0 a pole `Metric` má hodnotu 0xFF. Všechny RTE následující za takovým záznamem mají potom *Next Hop* určen z tohoto záznamu. Jako *Next Hop* adresa musí být adresa typu link-local, jinak je ignorována a *Next Hop* se opět určí ze zdrojové adresy zprávy (totéž v případě, kdy *Next Hop* = 0:0:0:0:0:0:0:0).

5.4 Zprávy typu `Response`

Zprávy typu `Response` obsahují dostupné sítě (připojené přímo nebo naučené pomocí protokolu RIPng) a jsou vytvořeny a odeslány pokud:

- je nutné vygenerovat pravidelnou aktualizaci (tzv. *Regular Update Message*) - generuje se každých 30 sekund viz kap. 5.6,
- se změnila metrika cesty (tzv. *Triggered Update Message*),
- je přijata zpráva typu `Request`.

5.4.1 Vytvoření zprávy typu `Response`

`Response` zpráva se vytváří pro každou připojenou síť (každé rozhraní), pokud se jedná o *Regular Update Message* nebo *Triggered Update Message*. Rozdíl mezi těmito zprávami je v jejich obsahu. *Regular Update Message* musí obsahovat všechny dostupné sítě, zatímco *Triggered Update Message* může obsahovat pouze RTE ze směrovací tabulky, u nichž se změnila metrika (např. síť se stala nedostupnou, tyto RTE mají nastaven *Route Change Flag*, viz kap. 5.7). Pro tyto typy zpráv je použita jako zdrojová adresa `link-local` adresa příslušného rozhraní (pokud má rozhraní více `link-local` adres, potom musí vybranou adresu používat po celou dobu, po které je dostupná), jako cílová adresa je nastavena multicastová skupina RIPng protokolu (`FF02::9`) a cílový port je RIPng port `521`. Na každé RTE v těchto zprávách je aplikován *split horizon* – síť se neodešle na rozhraní, skrze které byla naučena, viz kap. 3.

Pokud se vytváří `Response` zpráva, protože na ni byl přijat požadavek, odesílá se tato zpráva se zdrojem a cílem dle tabulky 5.1. Jestliže navíc požadavek obsahoval žádost na konkrétní RTE záznam (více o požadavcích v kapitole 5.5), tak v případě nalezení tohoto RTE záznamu ve směrovací tabulce, se na něj neaplikuje *split horizon*.

`Request` Message		`Response` Message			
Cíl. adr.	Zd. port	Zd. adr.	Zd. port	Cíl. adr.	Cíl. port
FF02::9	521	link-local	521	link-local	521
unicast	libovolný	global-unicast	521	`Request` zd. adr.	`Request` zd. port

Tabulka 5.1: Zdroj a cíl odpovědi na základě údajů požadavku

V `Response` zprávě se nesmí objevit RTE s `link-local` adresou.

Po odeslání *Regular Update Message* nebo *Triggered Update Message* jsou všechny *Route Change Flag* resetovány.

5.4.2 Zpracování zprávy typu `Response`

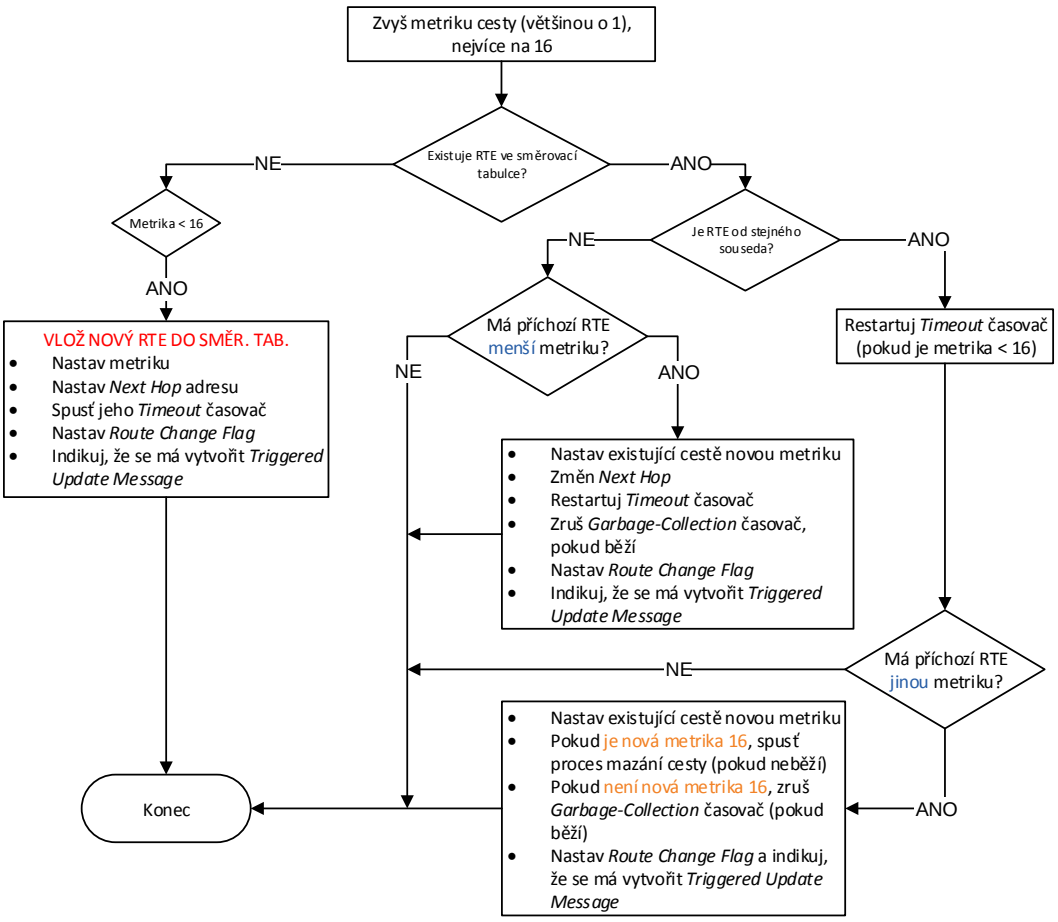
Při přijmutí odpovědi se kontroluje, zda:

- je z RIPng portu,
- je zdrojová adresa typu link-local,
- není zdrojová adresa vlastní,
- je u zpráv *Regular Update Message* nebo *Triggered Update Message* nastaven hop-count na 255 – zpráva přišla od souseda.

Pokud některá podmínka nevyhoví kontrole, je odpověď zahozena.

Po kontrole hlavičky zprávy se prochází všechny RTE. Každý záznam se zkontroluje, jestli obsahuje validní IPv6 prefix (nesmí být multicast, či link-local adresa), délku prefixu (0-128) a metriku (0-16). Pokud RTE neobsahuje zmíněné, je ignorován.

Každý RTE se zpracuje způsobem, který je znázorněn na obrázku 5.3.



Obrázek 5.3: Zpracování RTE ze zprávy typu `Response`

5.5 Zprávy typu `Request`

Zprávy typu `Request` obsahují požadavek na dostupné sítě a jsou většinou posílány, pokud:

- se spustil RIPng proces, který potřebuje zjistit všechny dostupné sítě;
- je potřeba diagnostikovat síť.

V prvním případě se požadavky odesílají jako multicast zpráva z RIPng portu. V případě druhém by měl být požadavek odeslán na unicastovou adresu směrovače a z jiného portu, než je RIPng port.

5.5.1 Zpracování zprávy typu `Request`

Zpráva typu `Request`, stejně jako zpráva typu `Response`, obsahuje RTE.

RTE v tomto případě znamená požadavek, zda je síť určená tímto RTE dostupná pro daný směrovač. Pro každou takovou síť se k RTE nastaví příslušná metrika. Pokud požadovaná síť není dostupná je metrika nastavena na 16. Typ zprávy se změní na `Response` a zpráva je odeslána zpět.

Pokud požadavek obsahuje pouze jediný záznam RTE, který má všechna pole, mimo pole metriky, nastavena na **0** a metriku nastavenu na **16**, znamená to požadavek na zaslání všech dostupných sítí (v odpovědi se aplikuje *split horizon*).

5.6 Časovače

RIPng používá následující časovače:

- Každých 30 sekund se generuje *Regular Update Message*, viz kap. 5.4.
- Každý RTE ve směrovací tabulce má vlastní *Timeout* časovač o délce 180 sekund. Po vypršení tohoto časovače se spustí proces mazání cesty – viz kap. 5.7.
- Každý RTE ve směrovací tabulce má navíc *Garbage-Collection Time* časovač, který se spouští při procesu mazání cesty. Tento časovač má délku 120 sekund a po jeho vypršení se RTE odstraní ze směrovací tabulky.
- *Triggered Update* časovač, jehož délka je určena náhodně od 1 do 5 sekund. Tento časovač se spouští před odesláním *Triggered Update Message* a pokud se má odeslat další *Triggered Update Message*, odešle se až po uplynutí tohoto časovače.

5.7 Proces smazání cesty

Tento proces je spuštěn pro RTE ve směrovací tabulce, pokud:

- mu vypršel *Timeout* časovač nebo
- se mu nastavila metrika o délce 16.

Proces smazání cesty zahrnuje následující akce:

- spuštění *Garbage-Collection Time* časovače pro daný RTE,
- nastavení metriky na 16,

- nastavení *Route Change Flag*,
- indikace, že se má vytvořit *Triggered Update Message* zpráva, viz kap. 5.4.

Kapitola 6

OMNeT++ a INET

Implementace protokolu RIPng má být v prostředí OMNeT++/INET, kterému se věnuje tato kapitola. Je zde stručně vysvětleno, co je OMNeT++ a jak pracuje. Dále je zde popsáno rozšíření (framework) INET, jenž implementuje některé funkce a protokoly pro simulaci počítačových sítí.

6.1 Prostředí OMNeT++

OMNeT++ [20] je simulační knihovna a prostředí pro vytváření simulací. Zaměřuje se především na simulace sítí (telekomunikačních, počítačových, senzorových atd.), ale díky své flexibilitě může být použit i např. pro simulaci rozsáhlých počítačových systémů nebo hardwarových architektur. Pro vytváření a zobrazení průběhu simulací poskytuje OMNeT++ grafické uživatelské prostředí (zobrazeno na obrázku 6.1).

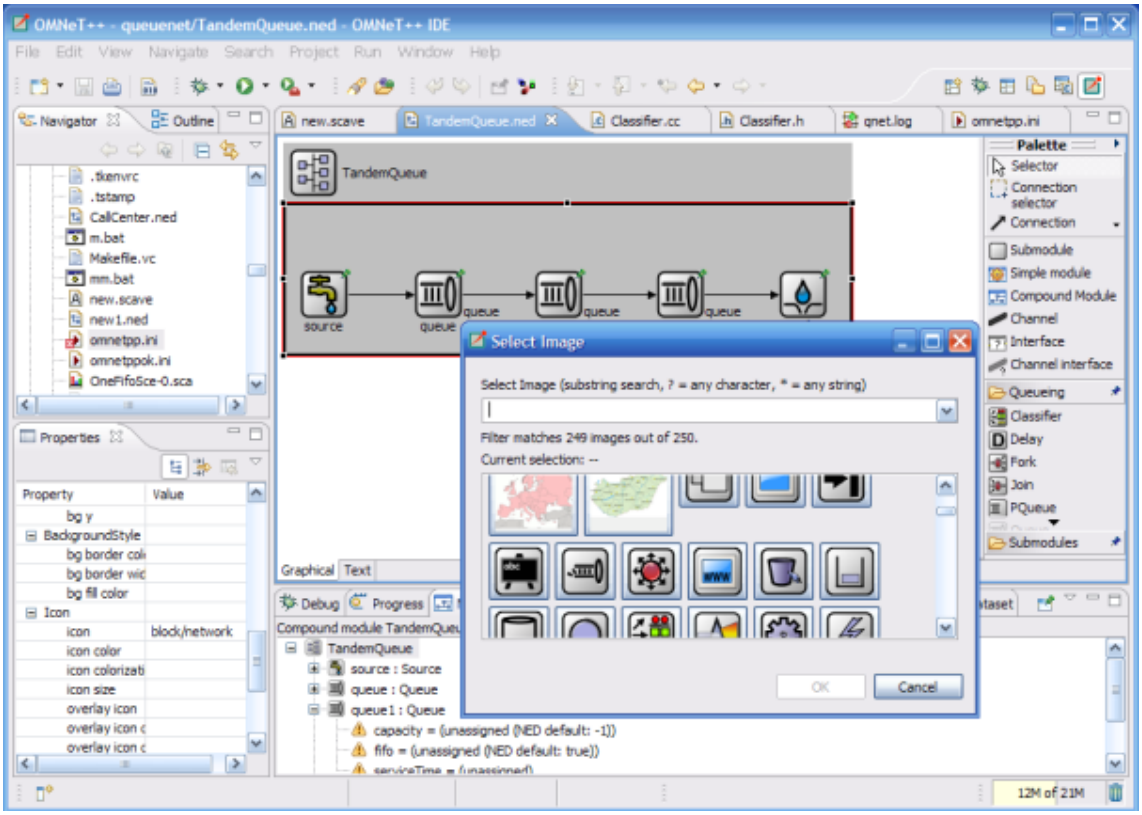
Prostředí OMNeT++ vzniklo jako studentský projekt na Technické Univerzitě v Budapešti v roce 1992. Za jeho autora je považován András Varga. První zveřejněná verze je z roku 1997. OMNeT++ je šířen jako software s otevřeným kódem pro akademické využití. Existuje také komerční verze, která se nazývá OMNEST[1]. V současné době je OMNeT++ stále ve vývoji a aktuální verze je 4.2.2.

6.1.1 Moduly

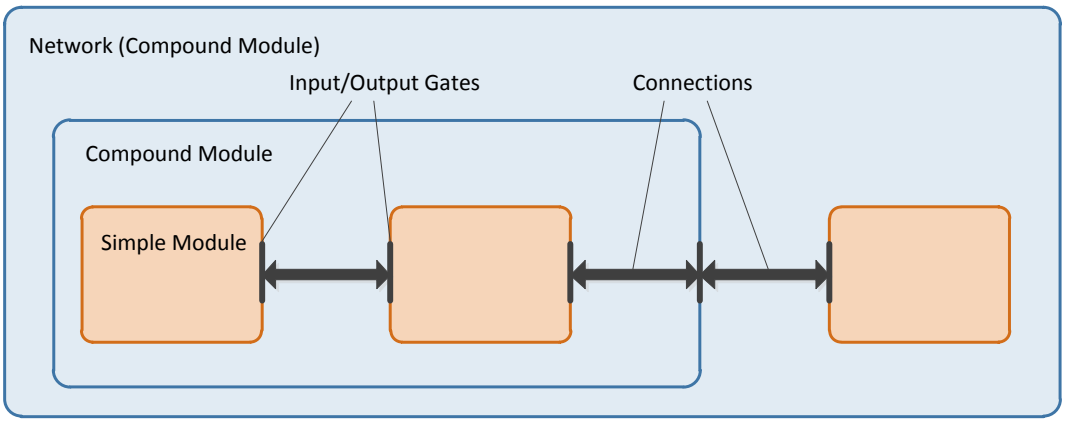
Simulační model je v OMNeT++ složen z hierarchicky uspořádaných modulů, které jsou znázorněny na obrázku 6.2. Na nejnižší úrovni jsou jednoduché moduly, *Simple Modules*, jež poskytují modelu funkcionalitu pomocí implementace metod v jazyce C/C++. Hierarchii pak vytvářejí složené moduly, *Compound Modules*, a kořenový modul se nazývá modul sítě, *Network Module*. Moduly spolu komunikují pomocí vstupních či výstupních bran (*input/output gates*), které jsou pomocí spojů (*connections*) propojeny ve složeném modulu. Přesný popis vytváření modulů lze nalézt v [21].

Každý modul může mít definované své parametry. Hodnotu parametru lze uvést přímo u jeho definice nebo v inicializačním souboru simulace `omnetpp.ini`, viz kapitola *Assigning Module Parameters* v [21].

[1] http://www.omnest.com/



Obrázek 6.1: Grafické uživatelské prostředí OMNeT++, převzato z [20]



Obrázek 6.2: Složení modelu v OMNeT++

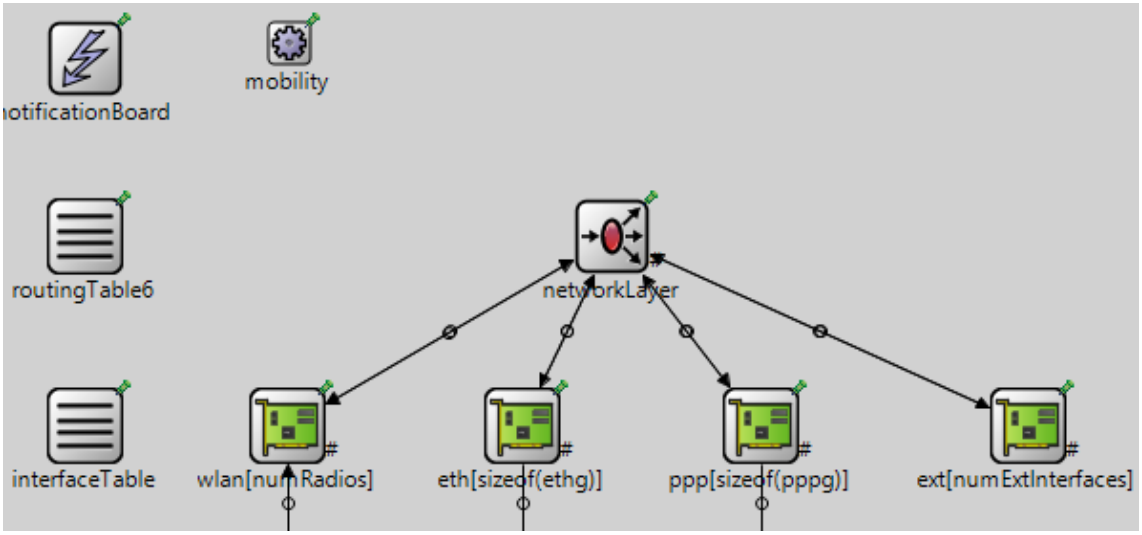
6.2 Rozšíření INET

INET Framework [11] je soubor knihoven (framework) pro prostředí OMNeT++, které poskytují funkce pro simulaci komunikačních, především počítačových, sítí. Obsahuje implementace protokolů z rodiny TCP/IP, ale také protokoly PPP, 802.11, MPLS nebo některé směrovací protokoly.

Přechůdce Frameworku INET byl balíček modulů IPSuite z roku 2000. V roce 2003 převzal IPSuite András Varga a všechny moduly přeorganizoval a zdokumentoval. V roce 2004 byl IPSuite přejmenován na INET Framework.

INET je v současné době stále ve vývoji s aktuální verzí 2.0.0.

Jak již bylo zmíněno, INET je rozšířením prostředí OMNeT++. Jednotlivé protokoly a části (rozhraní apod.) jsou tedy implementovány jako jednoduché, případně složené moduly, které můžou být dále použity pro vytvoření složitějšího celku, např. počítače, směrovače nebo celé počítačové sítě. Na obrázku 6.3 je zobrazena ukázka směrovače (`Router6.ned`) pro protokol IPv6, který je součástí Frameworku INET.



Obrázek 6.3: Ukázka směrovače v prostředí OMNeT++/INET

Kapitola 7

Současný stav Frameworku INET

V této kapitole je popsán současný stav implementace modulů směrovacích protokolů a modulů, které jsou potřebné pro implementaci protokolu RIPng ve Frameworku INET. Informace pro tuto kapitolu jsem čerpal zejména z manuálu [12] a dokumentace [13] pro INET.

7.1 Směrovací protokoly

Framework INET dlouhou dobu neobsahoval žádnou implementaci směrovacího protokolu pro počítačové sítě a v současné době obsahuje pouze implementaci OSPFv2[1] (modul `OSPFRouting`), která navíc není kompletní.

Framework INET dále rozšiřuje knihovna INETMANET[2], která implementuje ad-hoc směrovací protokoly:

- DSDV[3]
- AODV
- DSR[4]
- DYMO[5]

V rámci projektu ANSA [1] na Fakultě Informačních Technologií Vysokého Učení Technického v Brně vzniklo několik modulů, které rozšiřují Framework INET. Mezi těmito moduly je i směrovací protokol RIP verze 1 od Veroniky Rybové (viz [24]). Bohužel se tato verze protokolu RIP nepoužívá a navíc díky vývoji Frameworku INET, jehož autoři nehledí na zpětnou kompatibilitu, ji nelze přeložit a tedy ani použít v aktuální verzi knihovny INET.

[1] Open Shortest Path First - link-state směrovací protokol

[2] https://github.com/inetmanet/inetmanet

[3] DSDV - Destination sequenced distance vector routing: http://www.netlab.tkk.fi/opetus/s38030/k02/Papers/03-Guoyou.pdf, dnes se téměř nepoužívá a je nahrazen protokoly AODV nebo Babel

[4] DSR - Dynamic source routing protocol: http://en.wikipedia.org/wiki/Dynamic_Source_Routing

[5] DYMO nebo také jako návrh na standard AODVv2 je následník protokolu AODV: http://tools.ietf.org/html/draft-ietf-manet-dymo-25

7.2 UDP

Protokol RIPng využívá pro komunikaci protokol UDP. Implementace tohoto protokolu je v jednoduchém modulu `UDP`. Pokud chce jiný modul na aplikační vsrtvě komunikovat skrze modul UDP, musí mít definovanou vstupní bránu `udpIn` a výstupní bránu `udpOut` (brány mohou být pojmenovány jinak, ale je dobré dodržovat tuto konvenci) a tyto brány musí být propojeny s branami UDP modulu `appOut` a `appIn`.

Zaslání UDP datagramu je vhodné provést pomocí objektu typu `UDPSocket` (dále jako „socket“). Tomuto objektu se nejdříve nastaví výstupní brána, kterou má pro odesílání zpráv použít (např. `udpOut`) a další parametry u odesílaných datagramů (např. *Time To Live*, pomocí metody `setTimeToLive()` nebo zdrojovou adresu a port patřící danému socketu pomocí metody `bind()`). Datagram lze poté zaslat pomocí metody `sendTo()`.

Pokud modul zasílá zprávu pomocí socketu, musí mu nejprve přiřadit port (zdrojová adresa zprávy se nastaví automaticky na globální adresu odchozího rozhraní určeného při odesílání). Někdy je ale nutné u zasílaných zpráv uvádět linkovou adresu rozhraní. V tomto případě je možné vytvořit více „odchozích“ socketů a pomocí metody `bind(localAddr, localPort)` nastavit pro každé rozhraní správné zdrojové parametry.

Pokud má modul UDP přijímat zprávy z multicastové skupiny a ty dále předávat připojenému modulu na aplikační vrstvě, je nutné u socketu zavolat metodu `joinMulticastGroup()` s příslušnou multicastovou adresou. Pokud má modul více socketů, které používá pro komunikaci na lince, a pro všechny je zavolána zmíněná metoda, zpráva se pro každý takový socket duplikuje. To je ve většině případů nežádoucí. Řešením je vytvoření zvláštního socketu, který bude sloužit pro příjem multicastových zpráv.

7.3 IPv6

Přestože implementace protokolu IPv6 je ve vývoji a kód obsahuje spoustu zakomentovaných částí, které je nutné opravit, nebo části, u kterých je označeno, že je nutné je dokončit, je současná implementace dostačující pro zasílání unicastových i multicastových zpráv. Protokolu IPv6 ve Frameworku INET se detailně věnuje práce Marka Černého [7].

7.3.1 Směrovací tabulka

IPv6 směrovací tabulka je reprezentována modulem `RoutingTable6`. Cesty jsou v modulu uloženy ve vektoru a seřazeny podle délky prefixu. Hledání cesty je provedeno sekvenčně od nejdelšího prefixu.

Modul obsahuje implementované metody pro:

- vložení cesty pomocí `addRoutingProtocolRoute()` nebo `addStaticRoute()`,
- odebrání cesty – `removeRoute()`,
- získání cesty pomocí `getNumRoutes()` a `getRoute()`,
- vyhledání cesty s nejdelším shodným prefixem – `doLongestPrefixMatch()`,
- výpis směrovací tabulky – `info()`,
- oznámení změn ve směrovací tabulce (přídání/odebrání cesty) a

- optimalizaci (směrovací cache) apod.

Přístup k modulu směrovací tabulky není prostřednictvím bran a spojení, ale skrze třídu `RoutingTable6Access`.

Protože Framework INET dlouhou dobu neobsahoval funkční směrovací protokol (a pro IPv6 stále neobsahuje), funkce pro přidání cesty ve směrovací tabulce (nejen pro IPv6) nezohledňují cesty od různých směrovacích protokolů a umožňují přidat již vloženou cestu mající stejný prefix a délku prefixu.

7.3.2 IPv6 cesta

Cesty ve směrovací tabulce jsou reprezentovány třídou `IPv6Route`, ale je možné vytvořit vlastní třídu, která dědí z `IPv6Route` a do směrovací tabulky vkládat objekty této třídy. Toho je vhodné využít např. u RIPng – RTE může být pouze rozšířením `IPv6Route` a vkládán přímo do směrovací tabulky. Při obdržení cesty je možné pomocí C++ operátoru `dynamic_cast` ověřit, zda je cesta daného typu (patří k danému směrovacímu protokolu) a přetypovat ji.

Třída `IPv6Route` není přizpůsobena více běžícím směrovacím protokolům na jednom zařízení, protože neobsahuje atribut *administrativní vzdálenosti*, který musí být doplněn.

7.4 Konfigurace sítě

Konfigurovat zařízení (moduly reprezentující zařízení) v rozsáhlé síti pomocí parametrů prostředí OMNeT++ je velmi nepraktické a neflexibilní. Konfiguraci modulu lze ale zajistit pomocí členských funkcí jazyka C/C++, které nastavení načtou z lépe přizpůsobených souborů pro konfiguraci celé sítě. Framework INET je v této oblasti značně nekonzistentní, proto jsou v této kapitole popsány moduly, které jsou schopné načíst konfiguraci sítě. Pro automatickou konfiguraci sítě existují ve Frameworku INET následující moduly:

1. `FlatNetworkConfigurator`
2. `IPv4NetworkConfigurator`
3. `FlatNetworkConfigurator6`

Pro konfiguraci IPv6 sítě lze využít třetí modul, který pouze přiřazuje IPv6 adresy ze stejné sítě všem zařízením, případně spočítá nejkratší cesty mezi zařízeními a tyto cesty vloží do směrovací tabulky. První modul pak zastává stejnou funkci pro konfiguraci IPv4 zařízení. Pomocí druhéhou modulu je možné zařízení nakonfigurovat v různých (pod)sítích, ale také pouze v IPv4 adresovém prostoru.

Dále existuje třída `RoutingTableParser`. Pomocí ní lze ze souboru `irt`, který je svou strukturou podobný konfiguračním souborům známým z prostředí současných operačních systémů (cfg, ini, rc, atd.), např. načíst konfiguraci rozhraní nebo obsah směrovací tabulky. Tuto třídu využívá např. modul IPv4 směrovací tabulky (`RoutingTable`).

Některé moduly si ovšem načítají konfiguraci sami. Příkladem je modul směrovacího protokolu OSPFv2 (`OSPFRouting`), který načítá konfiguraci ze souboru ve formátu XML.

V rámci projektu ANSA byla vytvořena struktura konfiguračního souboru ve formátu XML, která má unifikovat nastavení zařízení. Tento soubor je navíc možné vytvořit automaticky z reálných aktivních síťových prvků, a poté ho využít pro analýzu sítě ve Frameworku

INET. Popisem a vytvářením uvedeného XML konfiguračního souboru se detailně zabývá diplomová práce Jakuba Smejkala [25] a v příloze A je uvedena část možné konfigurace. Přestože moduly využívající tento soubor ke konfiguraci si v současné době čtou nastavení pomocí svých funkcí, projekt ANSA klade důraz na přesun těchto funkcí do speciálního modulu (`DeviceConfigurator`), který bude obstarávat načtení konfigurace, a poté pomocí funkcí daného modulu provede jeho konfiguraci.

7.5 Události v síti

7.5.1 `ScenarioManager`

Pro plánování událostí v síti, jako je přerušení spojení nebo změna parametru zařízení, v současnoti existuje jediný modul – `ScenarioManager`. Tento modul umí pouze zrušit/vytvořit linku a změnit parametr danému modulu. `ScenarioManager` je nutné umístit přímo do *kořenového modulu* a pomocí parametru `script` mu nastavit XML soubor, ve kterém se nacházejí příkazy k provedení událostí. XML soubor má následující strukturu:

```
<scenario>
  <set-param t="10" module="host[1].mobility" par="speed" value="5"/>
  <at t="50">
    <set-param module="host[2].mobility" par="speed" value="10"/>
    <connect src-module="host[2]" src-gate="ppp[0]"
             dest-module="host[1]" dest-gate="ppp[0]"
             channel-type="ned.DatarateChannel">
      <param name="datarate" value="10Mbps" />
      <param name="delay" value="0.1us" />
    </connect>
  </at>
  <at t="60">
    <disconnect src-module="host[2]" src-gate="ppp[0]" />
  </at>
</scenario>
```

Každý příkaz musí mít parametr `t` určující čas jeho provedení. Příkazy uvedené v příkazu `at` nemusí parametr `t` obsahovat. Příkaz:

- `set-param` nastaví parametru `par` novou hodnotu `value` v rámci modulu `module`,
- `disconnect` zruší linku připojenou k rozhraní `src-gate` zařízení `module` a
- `connect` vytvoří novou linku typu `channel-type` (`channel-type` je modul popisující danou linku) mezi zařízeními `src-module` a `dest-module` s použitím rozhraní `src-gate` a `dest-gate`.

7.5.2 `InterfaceStateManager`

Modul `InterfaceStateManager` byl vytvořen v rámci projektu ANSA a slouží jako prostředník pro `ScenarioManager`. Lze pomocí něj ovládat stav rozhraní a rozšiřuje tak příkazy, které se můžou objevit v XML souboru pro `ScenarioManager`. Pokud `ScenarioManager` nedokáže interpretovat některý příkaz, zjistí, zda tento příkaz obsahuje atribut `module`. Tento

atribut slouží jako cesta k modulu, který implementuje rozhraní `IScriptable`, a kterému je příkaz předán. Modul `InterfaceStateManager` se umist'uje do modulu zařízení a dokáže interpretovat následující příkazy:

- `interfacedown`, pro vypnutí rozhraní `int` a
- `interfaceup`, pro zapnutí rozhraní `int`.

Příklad XML souboru pro InterfaceStateManager:

```
<scenario>
  <at t="60">
    <interfacedown module="R2.interfaceStateManager" int="eth1"/>
  </at>
  <at t="190">
    <interfaceup module="R2.interfaceStateManager" int="eth1"/>
  </at>
</scenario>
```

7.5.3 NotificationBoard

Pro upozornění modulů, že došlo k významné změně, slouží `NotificationBoard`. Upozornění může být např. na přidání cesty do směrovací tabulky nebo na změnu stavu rozhraní.

`NotificationBoard` se umist'uje do modulu zařízení a je přístupný skrze volaní C/C++ metod třídy `NotificationBoardAccess`.

Pokud nastane nějaká změna, patřičný modul informuje `NotificationBoard`, který dále informaci rozdistribuuje. Moduly, které chtějí být upozorněny na změny, musí implementovat rozhraní `INotifiable`, a poté můžou požádat `NotificationBoard` o upozornění na vybraný typ změny pomocí metody `subscribe()`.

Dostupné typy upozornění lze nalézt v hlavičkovém souboru `NotifierConsts.h`.

Kapitola 8

Návrh a implementace

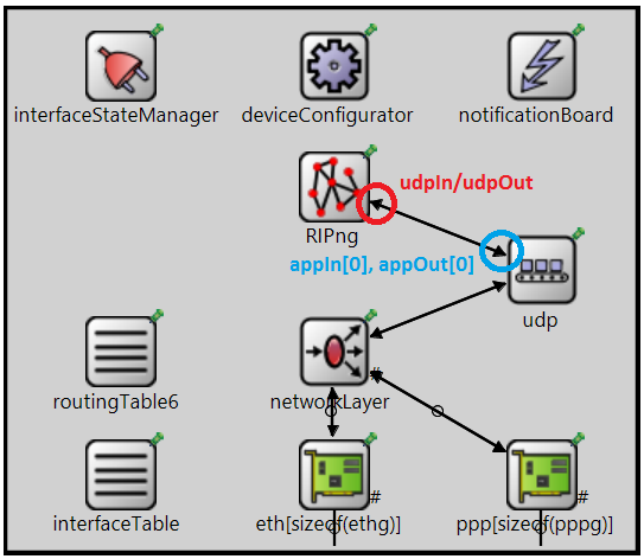
Protože návrh rozšíření Frameworku INET o protokol RIPng byl z větší části přímočarý, tato kapitola jej slučuje s implementací. Jsou zde popsány vytvořené struktury, jejich funkčnost a vztahy mezi nimi. Dále kapitola dokumentuje změny, které bylo nutné provést ve třídách a modulech Frameworku INET.

Struktury implementované pro protokol RIPng jsou ve jmenném prostoru `RIPng` nebo uvozeny odpovídající předponou (`RIPngRouting`, `RIPngProcess`...).

Diagram tříd a seznam souborů, které vznikly v rámci této práce lze nalézt v příloze B.

8.1 Modul protokolu RIPng

Protokol RIPng je reprezentován jednoduchým modulem `RIPngRouting`. Tento modul je možné vložit do směrovače a pomocí brány `udpIn` a `udpOut` jej propojit s modulem `UDP`. Pro ukázku funkčnosti byl vytvořen `RIPngRouter`, což je jednoduchý směrovač, který obsahuje pouze protokol RIPng a moduly potřebné pro jeho práci. Na obrázku 8.1 je zobrazena struktura tohoto směrovače – se zvýrazněným propojením modulů `RIPng` a `UDP` – v prostředí OMNeT++.



Obrázek 8.1: Modul `RIPngRouter` v prostředí OMNeT++/INET

Implementace modulu `RIPngRouting` je rozdělena do několika částí pro lepší přehlednost. Funkce jednotlivých částí jsou:

- Vytvoření a zrušení RIPng procesu (dále také jen „proces“). Proces je vytvořen/zrušen pomocí metod `addProcess()`/`removeProcess()`, kterým je předán jediný parametr – jméno procesu.
- Konfigurace procesu. Lze např. nastavit komunikační port a adresu nebo administrativní vzdálenost. Možnosti konfigurace a podporované příkazy jsou popsány v kapitole 8.5
- Povolení a zakázání RIPng na rozhraní s užitím metod `enableRIPngOnInterface()` (`disableRIPngOnInterface()`). Rozhraní je možné určit jeho jménem nebo pomocí ukazatele do tabulky rozhraní `InterfaceTable` (viz obr. 8.1).
- Vytváření, rušení a plánování časovačů jednotlivým procesům. Protože časovač v prostředí OMNeT++ znamená naplánování události a tyto události můžou plánovat pouze třídy dědící z `cSimpleModule` – tedy jednoduché moduly, není možné jej spravovat přímo v procesech. Metoda pro vytvoření časovače očekává mimo jiné také ukazatel na kontext, což je objekt, ke kterému se časovač vztahuje.
- Příjem a přeposlání zpráv. Zprávy protokolu RIPng musejí být odesílány s linkovou adresou a daným portem. K tomu je nutné vytvořit socket pro každou dvojici rozhraní–port, jak bylo popsáno v kapitole 7.2. Každý proces je také přiřazen ke „globálnímu“ socketu, který naslouchá na jeho portu a je přihlášen k dané multicastové skupině. Zprávy přijaté na tomto socketu jsou poté přeposlány příslušnému procesu (pokud na stejném portu a multicastové adrese naslouchá více procesů, je zpráva přeposlána pouze prvnímu z nich).

8.1.1 Inicializace

Po spuštění simulace si modul `RIPngRouting` nastaví parametry, které jsou definované v jeho `.ned` souboru (např. komunikační port a adresu). Těmito parametry také inicializuje všechny nově vytvořené procesy.

Ve druhém kroku se přihlásí k odběru notifikací, pro jejichž zpracování implementuje povinnou metodu `receiveChangeNotification()` z rozhraní `INotifiable` (viz kapitola 7.5.3). Aktuálně modul `RIPngRouting` dokáže reagovat na změnu stavu rozhraní (`NF_INTERFACE_STATE_CHANGED`) a obsahu směrovací tabulky (`NF_IPv6_ROUTE_DELETED`). Přijaté notifikace jsou distribuovány všem spuštěným procesům.

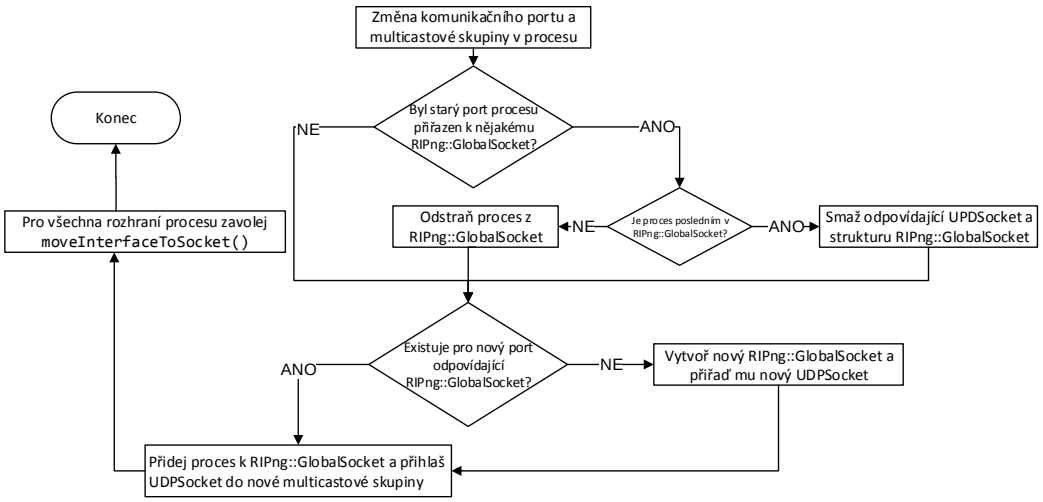
V posledním kroku se načte konfigurace ze souboru XML pomocí modulu `DeviceConfigurator`, do kterého byly přidány metody pro RIPng (popsány v kapitole 8.5). Teprve po přečtení celého konfiguračního souboru jsou spuštěny všechny požadované RIPng procesy.

8.1.2 Příjem a odesílání zpráv

Asociativní kontejner `sockets` obsahuje struktury typu `RIPng::Socket`. Každá tato struktura má v sobě `UPDSocket` určený pro odesílání zpráv a odkazy na rozhraní, které ho používají. Klíčem tohoto kontejneru je dvojice id-rozhraní a port, ke kterému je `UPDSocket` přihlášen. Ke správě `RIPng::Socket`ů je určena metoda `setOutputPortOnInterface()`.

Pro příjem zpráv je vytvořen asociativní kontejner `globalSockets` obsahující struktury typu `RIPng::GlobalSocket`. Typ `RIPng::GlobalSocket` obsahuje `UPDSocket`, který je určen k příjmu zpráv na daném portu (nebo pro odeslání paketu s globální unicastovou adresou), a odkazy na procesy, kreré jej využívají. Klíčem kontejneru `globalSockets` je port, ke kterému je socket přihlášen. Ke správě „globálních“ socketů je vytvořena metoda `moveProcessToSocket()`.

Metoda `moveProcessToSocket()` je znázorněna na obrázku 8.2 a je provedena po nastavení nového komunikačního portu a adresy procesu. Tato metoda se volá i pro nově vzniklé procesy, kde první podmínka skončí nepravdou. Podobně jako `moveProcessToSocket()` pracuje také `setOutputPortOnInterface()`.



Obrázek 8.2: Změna komunikačního portu a adresy RIPng procesu.

Vyhledání odpovídajícího socketu pro odeslání zprávy (a.) a procesu, kterému se má předat přijatá zpráva (b.) je znázorněno na obrázku 8.3. Použitá třída `RIPng::Interface` je popsána dále.

U odesílání stačí socket indexovat odchozím rozhraním a portem, který se má použít.

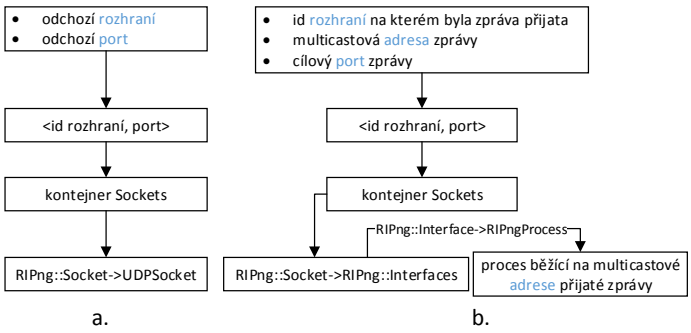
Přijatá zpráva se do modulu RIPng dostane pomocí „globálního“ socketu. Proces, kterému je určena, je možné určit pomocí socketu, který se používá pro odchozí datagramy na daném rozhraní a portu a poté nalézt první proces, který na zmíněném rozhraní používá určenou multicastovou adresu.

Na obrázku 8.4 je zobrazeno přijetí zprávy, které začíná v metodě `handleMessage()` (metoda implementovaná každým modulem, který chce přijímat zprávy – včetně oznámení časovačů).

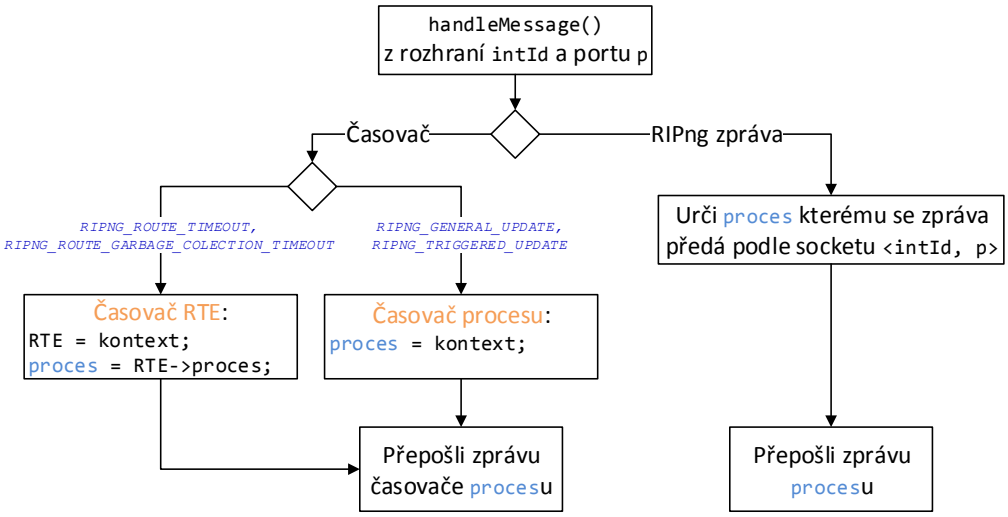
8.2 Třída `RIPngProcess`

Třída `RIPngProcess` reprezentuje samotný protokol RIPng. Při jejím návrhu jsem vycházel ze standardu RFC 2080, který je (stejně jako implementace) rozdělen do dvou hlavních částí, které se zabývají funkčností: *Input Processing* a *Output Processing*.

Třídu `RIPngProcess` jsem dále rozšířil o metody, které umožňují podobnou konfiguraci jako směrovače Cisco. Mimo chybějící podporu pro několik příkazů je větším rozdílem



Obrázek 8.3: Vyhledání socketu pro odeslání zprávy a procesu pro předání zprávy.



Obrázek 8.4: Přijetí zprávy v RIPngRouting modulu.

v chování absence *Holddown* časovače RTE záznamů. Přídání tohoto časovače by ale neměl být problém a je možné jej v budoucnu doplnit.

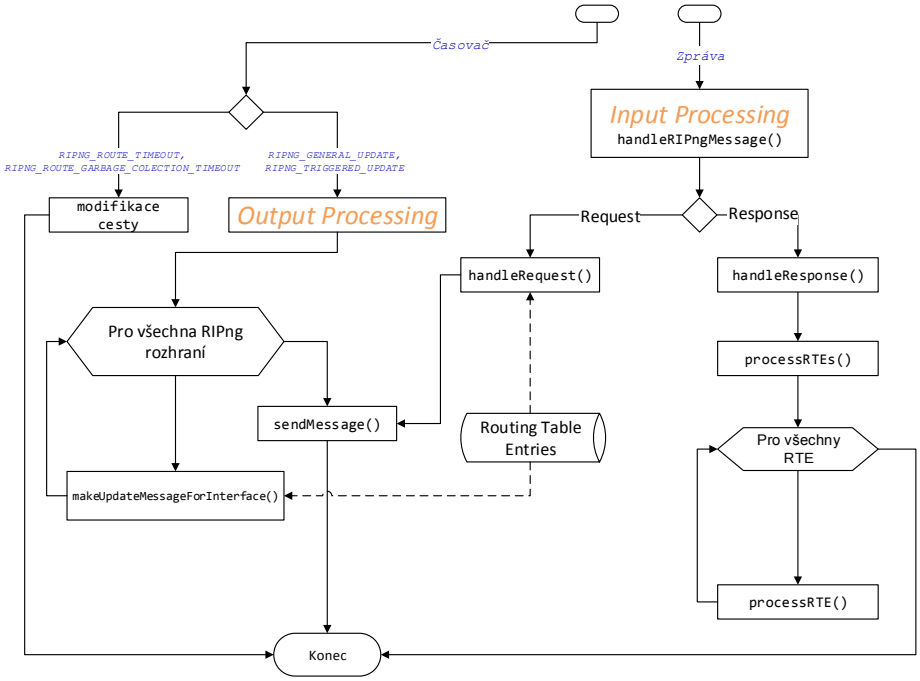
V implementaci stojí za povšimnutí použití výrazů `RTE` a `RoutingTableEntry`. Přestože oboje vyjadřuje totéž, zkratka `RTE` je používána výhradně pro označení záznamů, které se čtou (ukládají) ze (do) zpráv. `RoutingTableEntry` se pak používá při práci s RIPng databází. Pomocí uvedeného značení je možné lépe určit význam některých metod a proměnných.

Instanci RIPng procesu je nutno spustit metodou `start()`, ta vytvoří *Regular* a *Triggered Update* časovače. Nakonec odešle *Request* zprávu ze všech rozhraní, na kterých je proces povolen.

Celá aktivita objektu typu `RIPngProcess` po spuštění je znázorněna na obrázku 8.5 a popsána v následujících odstavcích.

Přijetí zprávy z modulu RIPngRouting je provedeno ve funkcích `handleTimer()` nebo `handleRIPngMessage()` podle toho, zda jde o časovač nebo RIPng zprávu.

Metoda `handleRIPngMessage()` rozpozná, zda se jedná o žádost nebo odpověď:



Obrázek 8.5: Aktivita třídy `RIPngProcess`, reprezentující RIPng protokol.

- Žádost je předána funkci `handleRequest()`, která zkontroluje počet RTE záznamů. V případě jedné RTE položky je zavolána metoda `getRTEs()`, která pro dané rozhraní získá všechny dostupné sítě podle jeho nastavených pravidel *split horizon* a *poison reverse*. V případě více RTE záznamů se dohledají pouze patřičné sítě. Odpověď je poté předána pomocí metody `sendMessage()` modulu `RIPngRouting`, který se postará o odeslání zprávy.

- Odpověď je nejprve zpracována funkcí `handleResponse()`. Ta zkontroluje zprávu jako celek (jestli má správný *hop-count* nebo je zdrojová adresa linková) a poté předá obsažené RTE záznamy ke zpracování do metody `processRTEs()`. Každé RTE je zkontrolováno, zda je validní a dále zpracováno funkcí `processRTE()`, jejíž implementace odpovídá algoritmu uvedeném v kapitole 5, ale cesta se navíc vkládá do směrovací tabulky zařízení (ne pouze jako ukazatel, ale jako její kopie).

Metoda `handleTimer()` určí, zda se zpracovává časovač pro RTE záznam ve směrovací tabulce nebo pro odeslání zprávy:

- Pro RTE záznam se v případě *Route Timeout* časovače spustí proces smazání cesty reprezentovaný funkcí `startRouteDeletionProcess()` stejně, jako je popsáno na konci kapitoly 5. Navíc se ale odstraní cesta ze směrovací tabulky zařízení, protože RIPng si udržuje vlastní databázi. Odeslání *Triggered Update* zprávy je zajištěno pomocí metody `sendDelayedTriggeredUpdateMessage()`, která nejdříve indikuje, že došlo ke změně v databázi, a poté odstartuje *Triggered Update* časovač (po jehož vypršení se zpráva odešlě). Tímto je zajištěno, že se negeneruje zpráva pro každou změnu zvlášť (většinou se více sítí stane nedostupných ve stejný čas). V případě *Garbage-Collection Time* časovače se cesta pouze odstraní z databáze.

- Pokud se má odeslat zpráva, je pro každé rozhraní vytvořena pomocí funkce `makeUpdateMessageForInterface()`, která využívá již popsaných metod `getRTEs()` a `sendMessage()`. Stejných metod, ale s jinými parametry, je využito pro odeslání pouze cest se změnou.

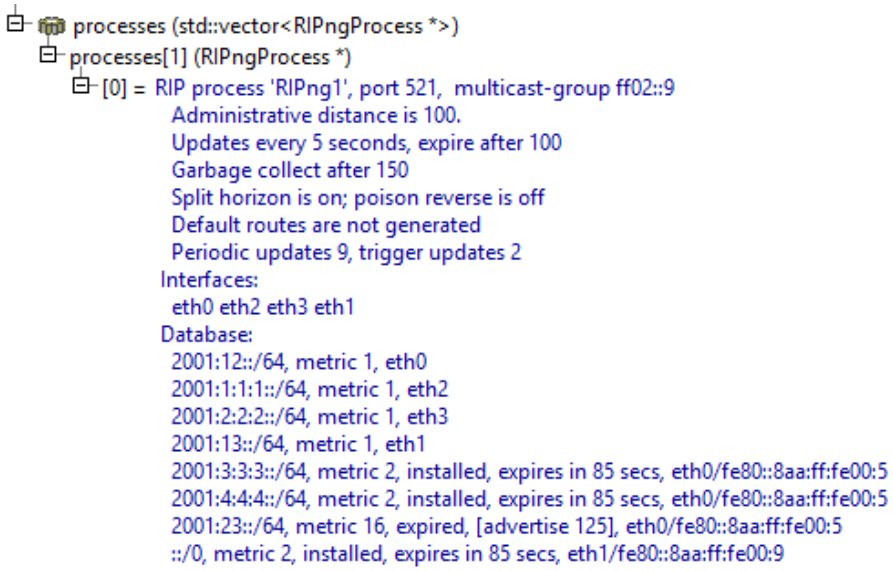
8.2.1 Notifikace

Notifikace je přijata RIPng modulem a předána procesu pomocí metody `handleNotification()`. Reakce na notifikace jsou následující:

- `NF_INTERFACE_STATE_CHANGED`
  - Vypnuté rozhraní – u všech cest s výstupním rozhraním, které vypadlo, se předpokládá, že jim vypršel *Timeout* časovač a je generována *Triggered Update* zpráva. Navíc je nutná operace s uloženým RIPng rozhraním, které je popsáno v kapitole 8.3.1.
  - Zapnuté rozhraní – odešle se *Triggered Update* zpráva s přímo připojenými sítěmi.
- `NF_IPv6_ROUTE_DELETED`
  - Smazána cesta jiného protokolu – pokud RIPng proces zná cestu do smazané sítě, pokusí se ji přidat do směrovací tabulky.

8.2.2 Zobrazení nastavení a databáze procesu

Při simulaci je možné v grafickém prostředí OMNeT++ rozkliknout modul `RIPngRouting`, ve kterém lze zobrazit parametry a databázi běžících procesů (položka `processes`). Zobrazení informací je omezené možnostmi OMNeT++, ale bylo navrženo po vzoru směrovačů Cisco (viz kapitola 4). Výpis procesu odpovídá příkazům `show ipv6 rip` *`process-name`* a `show ipv6 rip` *`process-name`* `database`. Na obrázku 8.6 je příklad tohoto výpisu.



Obrázek 8.6: Ukázka výpisu informací o procesech v RIPng modulu.

8.3 Struktury protokolu RIPng

8.3.1 `RIPng::Interface`

Každý proces má vlastní vektor (`enabledInterfaces`) s rozhraními, na kterých je povolen. Rozhraní RIPng procesu je reprezentováno třídou `RIPng::Interface`, ve které jsou uloženy potřebné informace: odkaz na „fyzické“ rozhraní zařízení, zda je povolen *split horizon*, *poison reverse* nebo jestli se má skrze něj šířit defaultní cesta.

Pokud dojde např. ke spadnutí linky, na které byl povolen RIPng proces, odpovídající RIPng rozhraní se přesune do vektoru `downInterfaces` (aby nedošlo ke ztrátě nastavených parametrů), ze kterého může být obnoveno, pokud linka začne opět operovat.

8.3.2 `RIPng::RoutingTableEntry`

Třída `RIPng::RoutingTableEntry` rozšiřuje třídu `ANSAIPv6Route` (popsána dále) a reprezentuje *Routing Table Entry* protokolu RIPng. Oproti `ANSAIPv6Route` přidává především potřebné časovače a metodu `RIPngInfo()`. Pomocí této metody jsou vypisovány informace o RIPng databázi podobně, jako na zařízeních Cisco.

V každém `RIPng::RoutingTableEntry` je navíc ukazatel na jeho kopii, která se vytvoří, pokud proces vkládá cestu do směrovací tabulky zařízení (cesty jsou tedy vzájemně svázané). Přímé vkládání `RIPng::RoutingTableEntry` do směrovací tabulky je možné díky dědičnosti jazyka C++.

8.3.3 `RIPngTimer`

Časovače se v prostředí OMNeT++ definují jako zprávy, proto byla vytvořena struktura RIPngTimer, která v sobě nese informaci o typu (*Regular Update* apod.). Navíc pomocí metody `setContext()` má každý časovač nastaven ukazatel na objekt (procesu nebo cesty), ke kterému se vztahuje.

8.3.4 `RIPngMessage`

Tento typ slouží k přenosu RIPng zpráv mezi směrovači. Jeho položky odpovídají standardu RFC 2080.

8.4 Úpravy ve třídách INET Frameworku

8.4.1 `UDP - ANSAUDP`

Modul UDP ve Frameworku INET neumožňuje následující (přestože opačné pořadí naslouchání problém nepůsobí):

```
UDPSocket socket1, socket2;
socket1.bind(RIPngPort);           //Vytvoř socket pro příjem zpráv
socket2.bind(address, RIPngPort);  //Socket pro odesílání
```

Je to způsobeno podmínkou v metodě `bind()`, která socketu zakazuje naslouchat na určeném portu a adrese, pokud jiný socket naslouchá pouze na zmíněném portu. Proto byla vytvořena třída `ANSAUDP`, která má benevolentnější `bind()` (používá funkci `ANSAfindSocketByLocalAddress()` namísto `findSocketByLocalAddress()`).

8.4.2 RoutingTable6 - ANSARoutingTable6

V rámci této práce byla vytvořena třída `ANSARoutingTable6`, která rozšiřuje třídu `RoutingTable6` (třída reprezentující směrovací tabulku), aby v zařízeních bylo možné spravovat více směrovacích protokolů. Po vzoru zařízení Cisco byla do všech cest přidána administrativní vzdálenost (viz další kapitola), a proto pro přidání nové cesty do směrovací tabulky musely být některé funkce upraveny nebo vytvořeny (pozn.: v dalším textu pojem „směrovací tabulka“ nebo pouze „tabulka“ identifikuje také třídu `ANSARoutingTable6` a „cestou“ je rovněž myšlena třída `ANSAIPv6Route`):

- `prepareForAddRoute()` - funkce slouží pro kontrolu, zda k určené cestě existuje ve směrovací tabulce cesta do stejné sítě s menší administrativní vzdáleností. Pokud ano, je vrácena hodnota `false`, která značí, že není možné novou cestu přidat. Jinak je cesta v tabulce smazána a vrácena hodnota `true`.

  Pro smazání cesty v této funkci byla vytvořena metoda `removeRouteSilent()`, která nepošle notifikaci o odstranění. Směrovací protokoly by měly reagovat na oznámení `NF_IPv6_ROUTE_DELETED` (jež generuje `removeRoute()`) a pokud znají cestu do stejné sítě, jako odstraněná cesta, měly by ji vložit do tabulky. V případě odstranění cesty ve funkci `prepareForAddRoute()` je toto chování nežádoucí. Pokud by přesto v této metodě bylo potřeba vyvolávat oznámení o smazání cesty, je možné vytvořit další notifikaci, při které by bylo zakázáno vkládat směrovací informace.

  Každý modul, který chce vložit novou cestu do směrovací tabulky, je povinnen nejdříve zavolat metodu `prepareForAddRoute()`. Důvodem oddělení této metody od funkcí pro vkládání cesty, a její zvláštní volání, je minimalizace změn oproti třídě `RoutingTable6`.

- Metody pro přidání cesty - všechny metody pro přidání cesty ze třídy `RoutingTable6` byly upraveny tak, aby vkládaly cesty s administrativní vzdáleností (`ANSAIPv6Route`).

- `RoutingTable6` používá pro směrování „cache paměť“. Cache se ale chybně nemaže (neaktualizuje) při manipulaci s cestou (pomocí existující metody `purgeDestCache()`). K podobné situaci docházelo při vypnutém rozhraní, do kterého se datagramy pomocí cache nesprávně směrovaly. Toto je ve třídě `ANSARoutingTable6` opraveno.

- `routeChanged()` - metoda byla doplněna po vzoru třídy `RoutingTable` a je volána objektem cesty (pokud je vložen do tabulky), jestliže se mu změní některý z parametrů (metrika apod.). Informace o změně je dále šířena pomocí `NF_IPv6_ROUTE_CHANGED` notifikace. Pokud je potřeba, jsou zneplatněny záznamy v cache v dané směrovací tabulce.

Zobrazení směrovací tabulky bylo upraveno po vzoru Cisco zařízení (příkaz `show ipv6 route`, viz kapitola 4). Na obrázku 8.7 je demonstrována nynější podoba.

8.4.3 IPv6Route - ANSAIPv6Route

Třída `ANSAIPv6Route` rozšiřuje třídu `IPv6Route`.

- Před vložením cesty do směrovací tabulky je navíc nutné nastavit její AD pomocí `setAdminDist()`. Administrativní vzdálenosti jsou definovány v `ANSAIPv6Route` a uvozeny písmenem *d*.

Obrázek 8.7: Vzhled směrovací tabulky `ANSARoutingTable6` v prostředí OMNeT++/INET.

- Směrovací protokoly musí také určit zdroj cesty pomocí `setRoutingProtocolSource()`. Zdroje jsou opět definovány v `ANSAIPv6Route` a uvozeny znakem *s*.
- Metoda `info()` je upravena tak, aby o každé cestě byla vypsána informace ve stylu zařízení Cisco (viz obrázek 8.7). Pokud je vytvořena třída, která dědí z `ANSAIPv6Route` a objekty této třídy jsou vkládány do tabulky, nesmí metodu `info()` přepisovat. Výpis by nebyl konzistentní. Pokud je funkce pro výpis informací o cestě potřeba, je pro ni nutné použít jiný název. Příkladem je metoda `RIPngInfo()` pro RIPng *Routing Table Entry* (kapitola 8.3.2).

Každá metoda měnící nějaký parametr cesty (neboli *setter*) má k sobě duální verzi končící *Silent*. Při změně parametru se totiž volá popsaná funkce `routeChanged()` ve směrovací tabulce (pokud je cesta do nějaké tabulky vložena) a generuje se notifikace. U „*Silent* metod“ k tomutu nedochází. Proto pokud se mění více hodnot cesty zároveň, je vhodné volat *Silent* verze (vyjma u posledního nastavovaného parametru), aby se zbytečně negenerovaly notifikace.

8.5 Konfigurace modulu `RIPngRouting`

8.5.1 `DeviceConfigurator` a `xmlParser`

Jak bylo zmíněno v kapitole 7.4, ANSA projekt dbá na unifikaci konfigurace modulů pomocí XML jazyka. Proto byl vytvořen `DeviceConfigurator`, ve kterém se shromažďují funkce pro čtení nastavení a soubor s tímto nastavením se pro zařízení specifikuje pomocí modulového parametru `configFile` (např.: `**.router1.configFile = "config.xml"`).

Pro `RIPngRouting` modul byly dopsány dvě funkce:

- `loadRIPngConfig()` - třída `RIPngRouting` získa při inicializaci přístup k `DeviceConfigurator` (pomocí `ModuleAccess<DeviceConfigurator>("deviceConfigurator").get()`) a zavolá jeho metodu `loadRIPngConfig()`, která z konfiguračního souboru nejdřívě načte nastavení jednotlivých procesů a poté pro každý proces seznam povolených rozhraní.

- `loadPrefixesFromInterfaceToRIPngRT()` - slouží pro načtení prefixů z rozhraní, na kterém se povolil RIPng proces.

Třída `xmlParser` implementuje podpůrné funkce pro čtení z XML souboru. Metody doplněné v rámci této práce obsahují předponu `RIPng`.

8.5.2 XML konfigurace

V této sekci jsou popsány XML tagy, pomocí kterých je možné konfigurovat RIPng modul. Při návrhu konfigurace RIPng modulu jsem vycházel z možností zařízení Cisco, které jsou popsány v tabulce uvedené na konci kapitoly 4. Příklad nastavení sítě, ve které se pro směrování používá RIPng protokol, je uveden v příloze C.

Tag	Význam tagu
`<RIPng name="`*process-name*`"/>`	Obsažený v `<Interface/>` spustí na daném rozhraní proces se jménem *process-name*. Pokud proces neexistuje, je vytvořen. Vložený pod `<Routing6/>` znamená příkaz k vytvoření procesu se jménem *process-name*.
Tagy uvedené v `<RIPng/>` pod `<Routing6/>`	
`<PoisonReverse>`*bool*`</PoisonReverse>`	Povolí/zakáže *Posion Reverse* mechanismus.
`<SplitHorizon>`*bool*`</SplitHorizon>`	Povolí/zakáže *split horizon* mechanismus.
`<Port>`*port*`</Port>`	Nastaví port pro komunikaci.
`<Address>`*multicast-addr*`</Address>`	Nastaví adresu pro komunikaci.
`<Distance>`*admin-dist*`</Distance>`	Změní administrativní vzdálenost.

`<Timers>`
`  <Udate>`*updatetimer*`</Update>`
`  <Route>`*route-timer*`</Route>`
`  <Garbage>`*garbage-timer*`</Garbage>`
`</Timers>`
	Změní délku časovačů *Regular Update*, *Timeout* cesty a *Garbage-Collection Time* cesty.

Tagy uvedené v `<RIPng/>` pod `<Interface/>`	
`<PassiveInterface>{`*disable*`|`*enable*`}` `</PassiveInterface>`	Nastaví rozhraní jako pasivní - nebudou se na něm odesílat RIPng aktualizace.
`<SplitHorizon>{`*disable*`|`*enable*`}` `</SplitHorizon>`	Vypne/zapne na rozhraní *split horizon*.
`<PoisonReverse>{`*disable*`|`*enable*`}` `</PoisonReverse>`	Vypne/zapne na rozhraní *poison reverse*.
`<MetricOffset>`*offset-value*`</MetricOffset>`	Upraví inkrementování metriky přijatých cest na daném rozhraní.

`<DefaultInformation>`
`  <Metric>`*metric-value*`</Metric>`
`  <DefaultOnly>`*bool*`</DefaultOnly>`
`</DefaultInformation>`
	Vloží IPv6 defaultní cestu (::/0) do aktualizací na daném rozhraní (a v určeném procesu). Pokud je uveden tag `<DefaultOnly/>` s hodnotou *true*, na tomto rozhraní se bude odesílat pouze defaultní cesta.

Tabulka 8.1: Přehled příkazů pro konfiguraci modulu `RIPngRouting`

Z tabulky 8.1 jsou patrné rozdíly oproti konfiguraci RIPng na zařízeních Cisco. Například *split horizon* a *poison reverse* lze konfigurovat také zvlášť pro rozhraní. V současné implementaci se nejdříve aplikuje nastavení procesu, poté každého rozhraní.

Příkaz `PassiveInterface` pro RIPng na zařízeních Cisco neexistuje. Důvodem zmíněného rozdílu je, že v Ciscu lze zavolat `redistribute connected`, a tím simulovat pasivní rozhraní. Funkce pro zpracování obdoby příkazu `redistribute` zatím nejsou v této práci vytvořeny.

Kapitola 9

Simulace

V této kapitole je popsáno několik testů, pomocí kterých byla ověřena implementace protokolu RIPng v prostředí OMNeT++/INET. Průběh simulace byl porovnán vůči běhu reálných zařízení Cisco 7206VXR směrovač s nainstalovaným operačním systémem C7200-ADVENTERPRISEK9-M verze 15.2(4)M2.

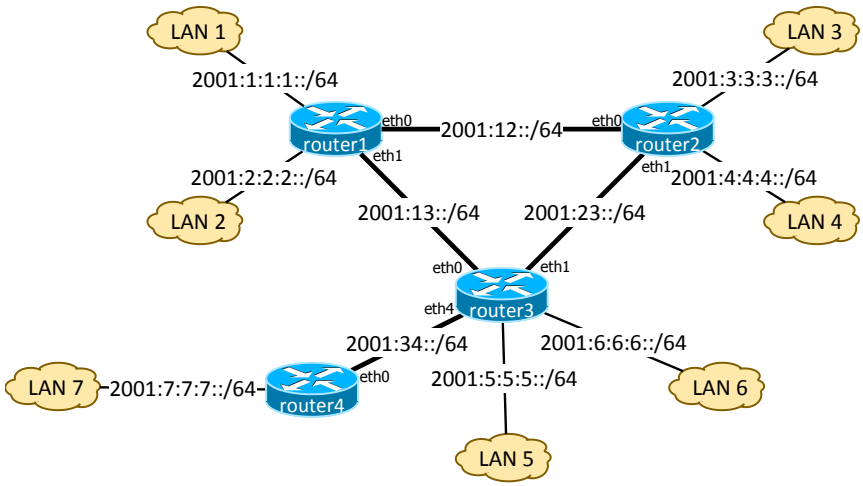
Protože OMNeT++ je diskrétní simulátor, časové průběhy simulace a reálné topologie nelze porovnávat přímo, ale pouze jako rozdíly mezi dobami odeslaných zpráv. Reálná i simulovaná zařízení mají jako T_0 označen čas, který se vztahuje k nějaké události (vždy uvedena v dané situaci).

Kapitola popisuje několik scénářů, které jménem odpovídají scénářům v inicializačním souboru simulace.

9.1 Topologie

Pro všechny scénáře byla použita topologie, která je znázorněna na obrázku 9.1. Konfigurace uzlů je uvedena v příloze C. LAN sítě byly u reálné topologie simulovány *loopback* rozhraními.

Rozhraní ve výpisech reálných zařízení jsou označeny jako FastEthernet1/*číslo-za-eth*.



Obrázek 9.1: Topologie sítě použitá k testování.

9.2 Scénář `RIPngTest1`

V tomto scénaři je ukázáno chování `RIPngRouter` směrovače po zapnutí – tj. generování *Request* zprávy a naplnění směrovací tabulky. V průběhu simulace bylo také naplánováno vypnutí rozhraní spojující zařízení *router2* a *router3* pro otestování reakce RIPng modulu na změnu topologie. Směrovač *router4* nebyl použit.

Po povolení protokolu na všech zařízeních zároveň, došlo k následujícím událostem:

1. Generování *Request* zpráv.
2. Odeslání odpovědí.
3. Nově naučené sítě jsou odeslány pomocí *Triggered Update* zpráv, konvergence sítě.
4. Zařízení si začnou vyměňovat *Regular Update* zprávy.

Na obrázku 9.2 je výpis komunikace, která byla odchycena pomocí programu Wireshark[1] mezi reálnými zařízeními na linkách *router1*–*router2*, *router1*–*router3* a *router2*–*router3*. Prvních devět řádků odpovídá událostem 1 a 2. Řádky deset až třináct události 3. Ostatní řádky značí *Regular Update* zprávy.

Jednotlivé sloupce v obrázku znamejí: číslo řádku, čas přijetí, zdrojová adresa, cílová adresa a typ zprávy.

1	0.148000	fe80::c802:8ff:fe64:1c	ff02::9	Command Request, Version 1
2	0.390000	fe80::c802:8ff:fe64:1d	ff02::9	Command Request, Version 1
3	0.645000	fe80::c801:8ff:fe64:1c	ff02::9	Command Request, Version 1
4	0.736000	fe80::c802:8ff:fe64:1c	fe80::c801:8ff:fe64	Command Response, Version 1
5	0.867000	fe80::c803:cff:fef4:1c	ff02::9	Command Request, Version 1
6	0.922000	fe80::c802:8ff:fe64:1d	fe80::c803:cff:fef4	Command Response, Version 1
7	1.150000	fe80::c803:cff:fef4:1d	ff02::9	Command Request, Version 1
8	1.351000	fe80::c801:8ff:fe64:1d	ff02::9	Command Request, Version 1
9	1.473000	fe80::c803:cff:fef4:1d	fe80::c801:8ff:fe64	Command Response, Version 1
10	2.123000	fe80::c801:8ff:fe64:1d	ff02::9	Command Response, Version 1
11	2.123000	fe80::c801:8ff:fe64:1c	ff02::9	Command Response, Version 1
12	5.976000	fe80::c802:8ff:fe64:1d	ff02::9	Command Response, Version 1
13	6.773000	fe80::c803:cff:fef4:1c	ff02::9	Command Response, Version 1
14	9.873000	fe80::c801:8ff:fe64:1d	ff02::9	Command Response, Version 1
15	9.921000	fe80::c801:8ff:fe64:1c	ff02::9	Command Response, Version 1
16	11.845000	fe80::c803:cff:fef4:1c	ff02::9	Command Response, Version 1
17	11.845000	fe80::c803:cff:fef4:1d	ff02::9	Command Response, Version 1
18	12.871000	fe80::c802:8ff:fe64:1c	ff02::9	Command Response, Version 1
19	17.977000	fe80::c802:8ff:fe64:1d	ff02::9	Command Response, Version 1
20	17.977000	fe80::c802:8ff:fe64:1c	ff02::9	Command Response, Version 1
21	38.945000	fe80::c801:8ff:fe64:1d	ff02::9	Command Response, Version 1
22	38.945000	fe80::c801:8ff:fe64:1c	ff02::9	Command Response, Version 1
23	40.287000	fe80::c803:cff:fef4:1c	ff02::9	Command Response, Version 1
24	40.287000	fe80::c803:cff:fef4:1d	ff02::9	Command Response, Version 1
25	45.341000	fe80::c802:8ff:fe64:1d	ff02::9	Command Response, Version 1
26	45.341000	fe80::c802:8ff:fe64:1c	ff02::9	Command Response, Version 1

router1:	router2:	router3:
eth0 - fe80::c802:8ff:fe64:1c	**eth0** - fe80::c801:8ff:fe64:1c	**eth0** - fe80::c803:cff:fe64:1c
eth1 - fe80::c802:8ff:fe64:1d	**eth1** - fe80::c801:8ff:fe64:1d	**eth1** - fe80::c803:cff:fe64:1d

Obrázek 9.2: Komunikace na síti po spuštění RIPng.

Porovnání časových známek zpráv mezi simulovanými a reálnými zařízeními ukazuje tabulka 9.1, kde T_0 = povolení RIPng procesu na *router1*.

Menší rozptyly mezi zprávami požadavků a odpovědí u sim. času jsou způsobeny nulovou časovou náročností operací v diskretní simulaci. Rychlost konvergence sítě, událost 3, je závislá na *Triggered Update* časovačích (jsou náhodně nastaveny od 1 do 5 sekund), po kterých se odesílají *Triggered Update* zprávy.

[1] Aplikace Wireshark slouží k zachytávání a analýze komunikace na síti – `http://www.wireshark.org`

Číslo události	Sim. čas [s]	Real. čas [s]
1	T_0	T_0-1.351
2	0.001	0.736-1.473
3	3.081-3.715	2.123-6.773
4	30+	9.873+

Tabulka 9.1: Porovnání časů přijatých a odeslaných zpráv RIPng protokolu.

Několik poznámek k obrázku 9.2:

- RIPng procesy na reálných zařízeních nelze spustit přesně ve stejný čas. Na některých rozhraních se tedy může vygenerovat *Request* zpráva dříve, než je spuštěn RIPng proces na rozhraní na opačném konci linky. Proto ve výpisu komunikace reálné sítě není ke všem *Request* zprávám odpovídající *Response* zpráva. To je také důvodem, proč zařízení *router1* odesílá ještě v čase 12.871 *Triggered Update* zprávu směrem k zařízení *router2* – *router1* neobdržel od *router3* odpověď na svůj požadavek na všechny sítě a tyto sítě se naučil teprve z *Regular Update* zprávy v čase 11.845.
- Z výpisu uvedené komunikace lze také vyčíst, že Cisco zařízení pro odeslání první *Regular Update* zprávy používají časovač o náhodné délce (jeho generování není uvedeno v žádné dostupné dokumentaci). V simulaci se první *Regular Update* zpráva odešle po 30 sekundách, což odpovídá *Regular Update* časovači (každé zařízení by po spuštění mělo znát všechny sítě pomocí *Triggered Update* zpráv v krátkém časovém intervalu).

Obrázek 9.3 ukazuje RIPng databázi simulovaného a reálného zařízení *router1* po 3. události (po dokončení konvergence sítě). Do všech sítí, které nejsou přímo připojeny existuje cesta. (Pozn.: databáze u reálného zařízení může obsahovat více cest do jedné sítě, jako je tomu např. u sítě 2001:23::/64. Tato funkce není u simulovaného zařízení v současné době podporována.)

```
Database:
 2001:12::/64, metric 1, eth0
 2001:1:1:1::/64, metric 1, eth2
 2001:2:2:2::/64, metric 1, eth3
 2001:13::/64, metric 1, eth1
 2001:3:3:3::/64, metric 2, installed, expires in 174 secs, eth0/fe80::8aa:ff:fe00:5
 2001:4:4:4::/64, metric 2, installed, expires in 174 secs, eth0/fe80::8aa:ff:fe00:5
 2001:23::/64, metric 2, installed, expires in 174 secs, eth0/fe80::8aa:ff:fe00:5
 2001:5:5:5::/64, metric 2, installed, expires in 174 secs, eth1/fe80::8aa:ff:fe00:9
 2001:6:6:6::/64, metric 2, installed, expires in 174 secs, eth1/fe80::8aa:ff:fe00:9
RIP process "RIPng1", local RIB
 2001:3:3:3::/64, metric 2, installed
     FastEthernet1/0/FE80::C800:FFF:FE08:1C, expires in 169 secs
 2001:4:4:4::/64, metric 2, installed
     FastEthernet1/0/FE80::C800:FFF:FE08:1C, expires in 169 secs
 2001:5:5:5::/64, metric 2, installed
     FastEthernet1/1/FE80::C802:FF:FEC4:1C, expires in 178 secs
 2001:6:6:6::/64, metric 2, installed
     FastEthernet1/1/FE80::C802:FF:FEC4:1C, expires in 178 secs
 2001:12::/64, metric 2
     FastEthernet1/0/FE80::C800:FFF:FE08:1C, expires in 169 secs
 2001:13::/64, metric 2
     FastEthernet1/1/FE80::C802:FF:FEC4:1C, expires in 178 secs
 2001:23::/64, metric 2, installed
     FastEthernet1/0/FE80::C800:FFF:FE08:1C, expires in 169 secs
     FastEthernet1/1/FE80::C802:FF:FEC4:1C, expires in 178 secs
```

Obrázek 9.3: RIPng databáze směrovače *router1*.

V průběhu simulace bylo náplánováno vypnutí rozhraní spojující *router2* a *router3*. Reakcí na tuto změnu v zařízení *router3* bylo odstranění sítí s odchozím rozhraním `eth1` a generování *Triggered Update* zprávy (ke stejným událostem došlo na reálném zařízení).

Sítě `2001:3:3:3::/64` a `2001:4:4:4::/64` (připojené k *router2*) se tedy pro *router3* stanou nedostupnými. Ale poté, co *router3* přijme první *Regular Update* zprávu od směrovače *router1*, nainstaluje si sítě připojené k *router2* s odchozím rozhraním *eth0*. Viz obrázek 9.4.

```
2001:3:3:3::/64, metric 3, installed, expires in 150 secs, eth0/fe80::8aa:ff:fe00:18
2001:4:4:4::/64, metric 3, installed, expires in 150 secs, eth0/fe80::8aa:ff:fe00:18
2001:3:3:3::/64, metric 3, installed
    FastEthernet1/0/FE80::C801:8FF:FED0:1D, expires in 166 secs
2001:4:4:4::/64, metric 3, installed
    FastEthernet1/0/FE80::C801:8FF:FED0:1D, expires in 166 secs
```

Obrázek 9.4: Cesty na *router3* po vypnutí rozhraní `eth1`.

9.3 Scénář `RIPngTest2`

V předchozím scénáři bylo ukázáno, co se stane při vypnutí rozhraní. RIPng protokol na tuto změnu reaguje ihned, a to generováním *Triggered Update* zprávy obsahující cesty s nekonečnou metrikou, které měly zmíněné rozhraní jako výstupní.

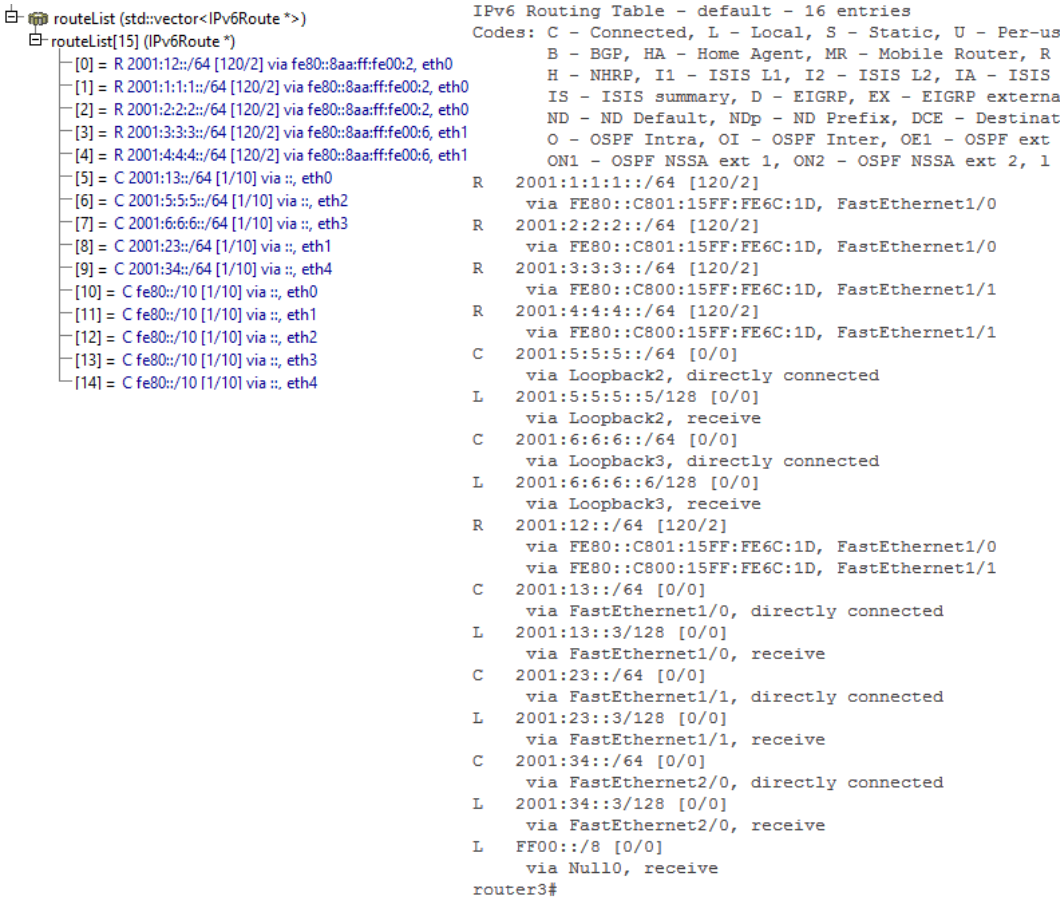
V tomto scénáři jsou otestovány *Timeout* časovače pomocí umlčení RIPng procesu na směrovači *router4*.

V čase T_0 *router4* odešle poslední *Regular Update* zprávu. Tabulka 9.2 ukazuje síť 2001:7:7:7::/64 v databázi protokolu RIPng na zařízení *router3*. (Pozn.: v tomto scénáři bylo pro aktualizaci výpisu databáze v simulaci naplánovány dva časovače $T_0 + 179$ a $T_0 + 181$ v inicializaci RIPng procesu na směrovači *router3*.)

Čas [s]	Stav cesty do sítě 2001:7:7:7::/64 v databázi RIPng	
	Reálné zařízení	Simulované zařízení
T_0	Poslední aktualizace *Timeout* časovače.	
179	2001:7:7:7::/64, metric 2, installed FastEthernet2/0/FE80::C805:17FF:FE9C:1C, expires in 1 secs	2001:7:7:7::/64, metric 2, installed, expires in 1 secs, eth4/fe80::8aa:ff:fe00:e
181	2001:7:7:7::/64, metric 16, expired, [advertise 119/hold 0] FastEthernet2/0/FE80::C805:17FF:FE9C:1C	2001:7:7:7::/64, metric 16, expired, [advertise 119], eth4/fe80::8aa:ff:fe00:e
300+	Cesta odstraněna.	

Tabulka 9.2: Reakce na vypnutý RIPng protokol na zařízení *router4*.

Obrázek 9.5 ukazuje směrovací tabulky na zařízeních *router3* v čase T_0 + 181s. Lze vidět, že cesta do sítě 2001:7:7:7::/64 byla odstraněna.



Obrázek 9.5: Směrovací tabulka simulovaného | reálného zařízení *router3* po vypršení *Timeout* časovače cesty do sítě 2001:7:7:7::/64.

9.4 Scénář `RIPngTest3`

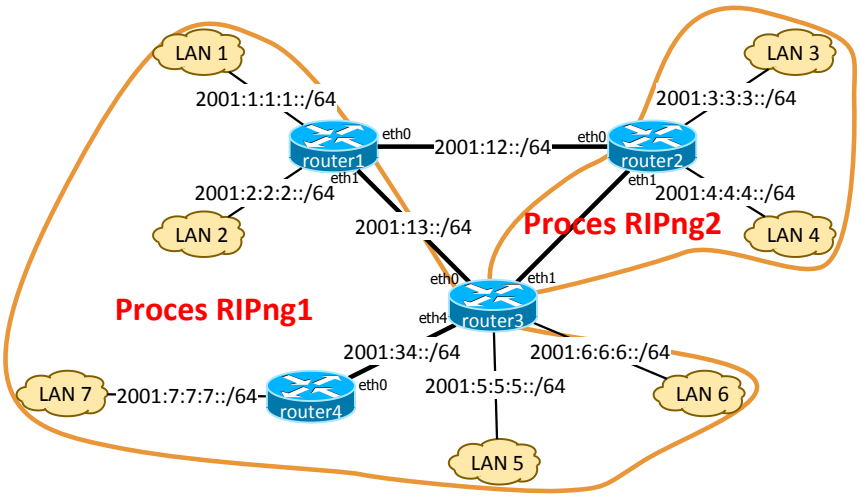
Ve scénáři `RIPngTest3` je síť rozdělena na dvě domény pomocí různých RIPng procesů na směrovači *router3* (mezi *router1* a *router2* neběží žádný proces) – viz obrázek 9.6. Zařízení *router1* tedy nezná sítě připojené k *router2* a naopak. Ale pomocí defaultní cesty šířené ze všech rozhraní na *router3* je možná komunikace mezi libovolnými LAN sítěmi.

Směrovač *router3* skrze rozhraní `eth0` šíří pouze defaultní síť ::0/0, takže *router1* má uloženou jedinou cestu – mimo přímo připojených sítí (obrázek 9.7a).

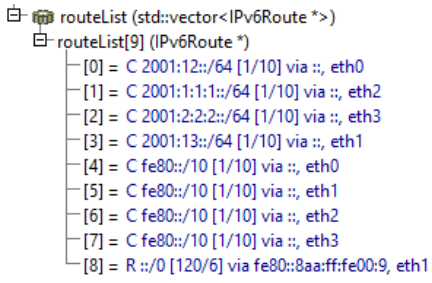
RIPng proces na směrovači *router2* běží pouze na rozhraní `eth1`. Ve směrovací tabulce je tedy pouze defaultní cesta (mimo přímo připojené sítě) od procesu `RIPng2` (na zařízení *router3*), protože ten běží na jediném rozhraní `eth1` (obrázek 9.7b).

Směrovač *router3* má informace o všech sítích a vystupuje jako centrální prvek (obrázek 9.7c).

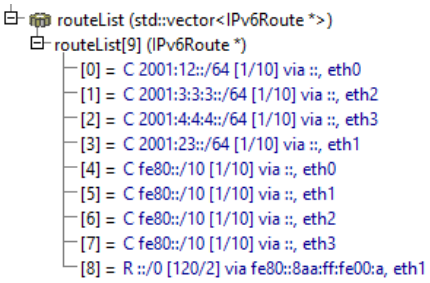
Pokud RIPng odesílá na některém rozhraní informaci o defaultní cestě, v příchozích zprávách je ignorována (pro zamezení směrovacích smyček). To je ukázáno na zařízení *router4*, které má „chybně“ nastavené šíření defaultní cesty, čehož důsledkem je, že síť ::0/0 od směrovače *router3* je ignorována a není obsažena ve směrovací tabulce (obrázek 9.7d).



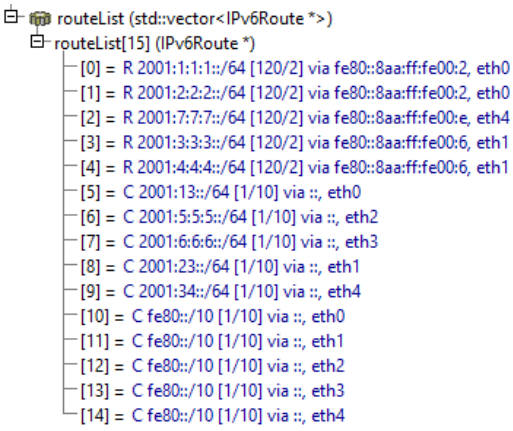
Obrázek 9.6: Rozdělení sítě pomocí dvou RIPng procesů na směrovači *router3*.



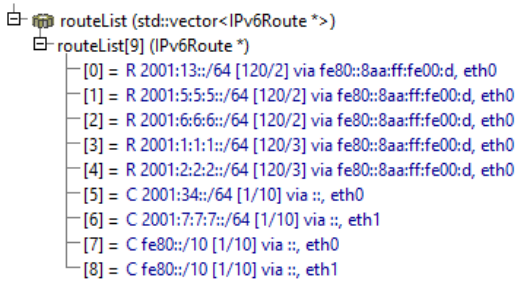
(a) *router1*



(b) *router2*



(c) *router3*



(d) *router4*

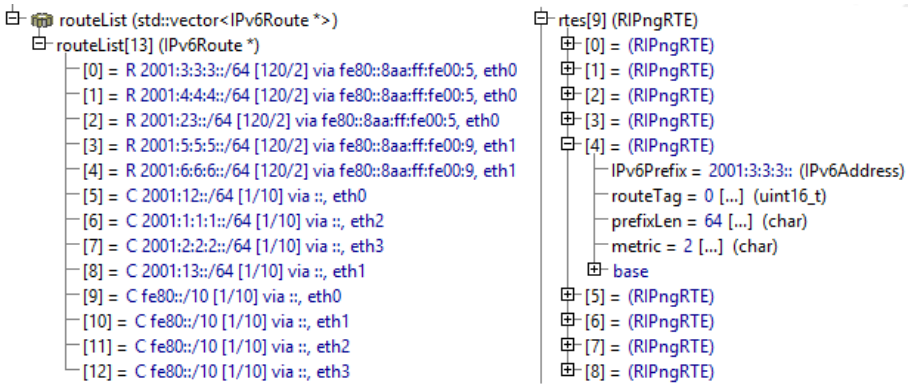
Obrázek 9.7: Scénář 3 - směrovací tabulky

9.5 Scénář `RIPngTest4`

V tomto scénáři byla otestována konfigurace RIPng modulu. Každá podkapitola popisuje nastavení parametru, který nebyl otestován v rámci předchozích scénářů a je uveden v tabulce na konci kapitoly 8.

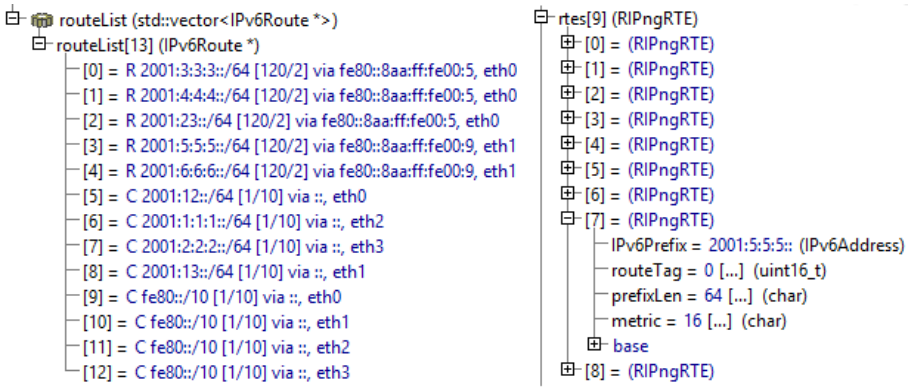
9.5.1 *Split horizon* a *poison reverse*

Rozhraní `eth0` na směrovači *router1* bylo nakonfigurováno s vypnutým *split horizon* mechanismem. Na obrázku 9.8 lze vidět, že přestože cesta do sítě 2001:3:3:3::/64 má jako výstupní rozhraní `eth0`, je skrze toto rozhraní šířena s metrikou 2.



Obrázek 9.8: *router1* - směrovací tabulka | zpráva odeslaná na rozhraní `eth0` s vypnutým *split horizon* mechanismem.

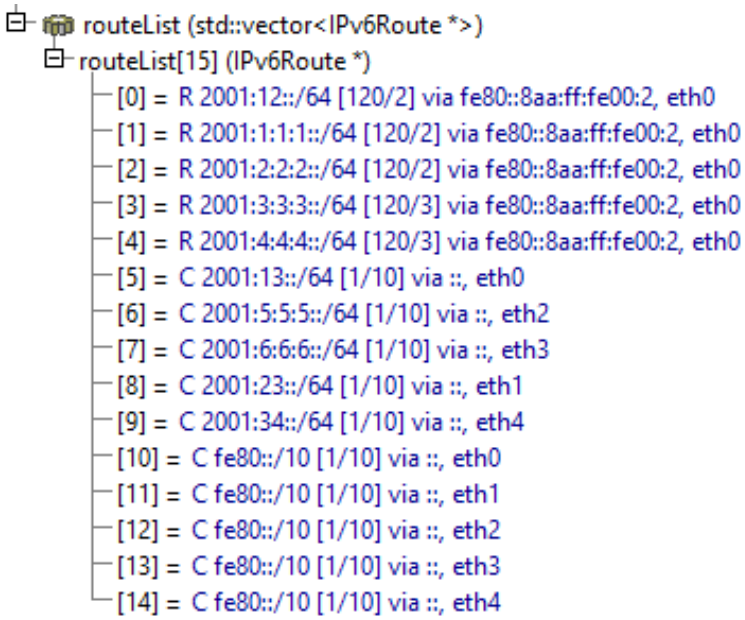
Podobně na rozhraní `eth1` je šířena cesta do sítě 2001:5:5:5::/64, ale s metrikou 16, protože na tomto rozhraní byl povolen mechanismus *poison reverse*. Viz obrázek 9.9.



Obrázek 9.9: *router1* - směrovací tabulka | zpráva odeslaná na rozhraní `eth1` se zapnutým *poison reverse* mechanismem.

9.5.2 *Metric-Offset*

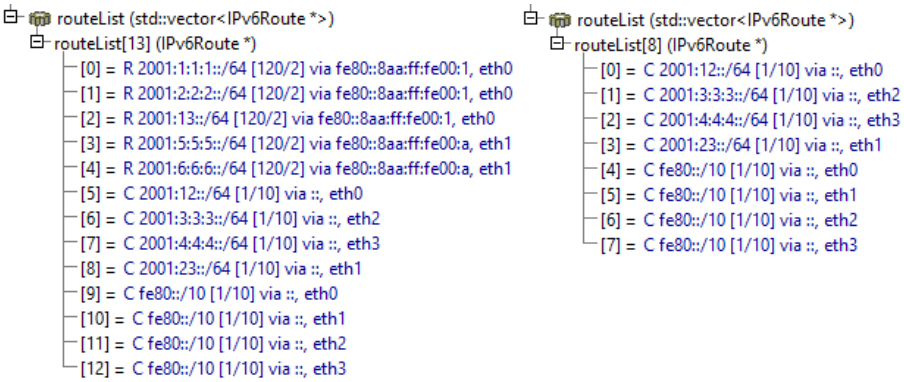
Na obrázku 9.10 je zobrazena směrovací tabulka směrovače *router3*. Přestože nejkratší cesta do sítí 2001:3:3:3::/64 a 2001:4:4:4::/64 je přes rozhraní `eth1`, je u nich uvedeno výstupní rozhraní `eth0`. Toto je způsobeno inkrementováním metriky o tři na rozhraní `eth1`.



Obrázek 9.10: *router3* - směrovací tabulka při zvýšení inkrementace metriky o dva na rozhraní `eth1`.

9.5.3 Délka časovačů

Na směrovači *router2* byla upravena délka časovačů: *Regular Update* na 10s, *Timeout* cesty na 3s a *Garbage-Collection Time* na 2s. Po zapnutí simulace se scénářem `RIPngTest4`, lze ve směrovací tabulce zařízení *router2* vidět, že cesty díky zkráceným časovačům nejsou stálé, protože ostatní zařízení mají defaultní hodnoty časovačů (*Regular Update* o délce 30s) – viz obrázek 9.11, na kterém jsou směrovací tabulky zařízení *router2* v čase přijetí *Regular Update* zpráv a o deset sekund později.



Obrázek 9.11: Nesprávně nastavené časovače. Směrovací tabulky zařízení *router2* v čase přijetí *Regular Update* zpráv a o deset sekund později.

Kapitola 10

Závěr

V této práci je popsána implementace protokolu RIPng v prostředí OMNeT++/INET.

RIPng je směrovací protokol určen pro IPv6 sítě, proto je v první části práce popsána IPv6 adresace. Jsou zde vysvětleny pojmy jako linková nebo multicastová adresa.

Následuje popis distance-vektor směrovacích protokolů a jejich příklady. Práce se zaměřuje na protokol RIPng, jehož specifikací se zabývá kapitola 5 a jako reference je uvedena konfigurace RIPng na směrovači Cisco.

Dále je provedena analýza prostředí OMNeT++ a Frameworku INET, které dohromady poskytují funkce pro simulování síťové komunikace. Je zde popsána současná implementace protokolů UPD a IPv6, směrovací tabulky, funkcí pro konfiguraci sítě a plánování událostí v síti v INET Frameworku. Všechny tyto části jsou stěžejní pro implementaci protokolu RIPng.

Samotnou implementací protokolu RIPng se zabývá kapitola 8. V této kapitole jsou popsány funkce a struktury, jež byly vytvořeny a které dohromady tvoří modul `RIPngRouting` reprezentující směrovací protokol RIPng. Implementace vychází ze standardu RFC 2080 a je upravena po vzoru směrovačů firmy Cisco.

Pro otestování implementace bylo vytvořeno několik scénářů. Výstup těchto scénářů byl také porovnán s výsledky získaných z reálné topologie.

Modul `RIPngRouting` nabízí možnost podobné konfigurace jako směrovače Cisco. Ale zpracování některých nastavení bylo nad rámec této práce. V budoucnu je však možné vytvoření funkcí, které budou vykonávat redistribuci cest z jiných směrovacích protokolů nebo sumarizaci cest na rozhraní.

V současné době je práce zahrnuta do ANSA projektu. Cílem je integrace modulu `RIPngRouting` do knihovny INET samotnými autory, což umožní přístup širší veřejnosti k tomuto rozšíření.

Literatura

[1] Automated Network Simulation and Analysis. ANSA, 2012, [Online; navštíveno 30.8.2012].
URL http://nes.fit.vutbr.cz/ansa/pmwiki.php

[2] Chroboczek, J.: The Babel Routing Protocol. RFC 6126, Internet Engineering Task Force (IETF), 2011.
URL http://www.ietf.org/rfc/rfc6126.txt

[3] Cisco Feature Navigator. Cisco, © 1992-2008, [Online; navštíveno 8.1.2012].
URL http://tools.cisco.com/ITDIT/CFN/jsp/by-feature-technology.jsp

[4] Implementing RIP for IPv6. Cisco, © 2001-2011, [Online; navštíveno 8.1.2012].
URL http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-rip.html

[5] Cisco: Deploying IPv6 in Unified Communications Networks with Cisco Unified Communications Manager 8.0(x). Technická zpráva, Cisco Systems, Inc., 2010.
URL http://www.cisco.com/en/US/docs/voice_ip_comm/cucm/srnd/ipv6/ipv6srnd.pdf

[6] Distance Vector Routing Protocols. Cisco, 2010, [Online; navštíveno 28.12.2012].
URL https://learningnetwork.cisco.com/servlet/JiveServlet/download/8472-1-12383/Distance%2520Vector%2520Routing%2520Protocols.doc

[7] Černý, M.: *Modelování IPv6 v prostředí OMNeT++*. master's thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2011.

[8] Ford, L. R. J.; Fulkerson, D. R.: *Flows in Networks*. Princeton University Press, 1962, ISBN 9780691146676.

[9] Hedrick, C.: Routing Information Protocol. RFC 1058, Internet Engineering Task Force (IETF), 1988.
URL http://www.ietf.org/rfc/rfc1058.txt

[10] Hinden, R.; Deering, S.: Internet Protocol Version 6 (IPv6) Addressing Architecture. RFC 3513, Internet Engineering Task Force (IETF), 2003.
URL http://www.ietf.org/rfc/rfc3513.txt

[11] Welcome to the INET Framework! INET FRAMEWORK, 2012, [Online; navštíveno 30.8.2012].
URL http://inet.omnetpp.org/

[12] Manual. INET FRAMEWORK, 2012, [Online; navštíveno 6.1.2012].
URL http://inet.omnetpp.org/doc/INET/inet-manual-draft.pdf

[13] release 2.0.0-096ccd0. INET FRAMEWORK, 2012, [Online; navštíveno 6.1.2012].
URL http://inet.omnetpp.org/doc/INET/neddoc/index.html

[14] Enhanced Interior Gateway Routing Protocol. Cisco, 2005, [Online; navštíveno 30.12.2012].
URL http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml

[15] An Introduction to IGRP. Cisco, 2005, [Online; navštíveno 30.12.2012].
URL http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a00800c8ae1.shtml

[16] IPv6 Multicast Address Space Registry. Stig Venaas, 2012-11-20, [Online; navštíveno 7.1.2012].
URL http://www.iana.org/assignments/ipv6-multicast-addresses/ipv6-multicast-addresses.xml

[17] Loshin, P.: *IPv6*. Morgan Kaufmann Publishers, 2004, ISBN 1-55860-810-9.

[18] Malkin, G.: RIPng for IPv6. RFC 2080, Internet Engineering Task Force (IETF), 1997.
URL http://www.ietf.org/rfc/rfc2080.txt

[19] Malkin, G.: RIP Version 2. RFC 2453, Internet Engineering Task Force (IETF), 1998.
URL http://www.ietf.org/rfc/rfc2453.txt

[20] OMNeT++. OMNeT++, Copyright© 2001-2012, [Online; navštíveno 30.8.2012].
URL http://www.omnetpp.org/

[21] User Manual. OMNeT++, Copyright© 2001-2012, [Online; navštíveno 5.1.2013].
URL http://www.omnetpp.org/doc/omnetpp/manual/usman.html

[22] Perkins, C.: Ad hoc On-Demand Distance Vector (AODV) Routing. RFC 3561, Internet Engineering Task Force (IETF), 2003.
URL http://www.ietf.org/rfc/rfc3561.txt

[23] Postel, J.: User Datagram Protocol. RFC 768, Internet Engineering Task Force (IETF), 1980.
URL http://www.ietf.org/rfc/rfc768.txt

[24] Rybová, V.: *Modelování a simulace návrhových vzorů směrování v počítačových sítí.* bachelor's thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2009.

[25] Smejkal, J.: *Dynamický stav modelu OMNeT++ pomocí SNMP.* master's thesis, Vysoké učení technické v Brně, Fakulta informačních technologií, 2012.

[26] University of Southern California: Internet Protocol. RFC 791, Internet Engineering Task Force (IETF), 1981.
URL http://www.ietf.org/rfc/rfc791.txt

Přílohy

Příloha A

Konfigurační XML soubor

Část možné konfigurace, převzato a upraveno z [25].

```
<Routers>
  <Router id=("id")>
    <Hostname></Hostname>
    <Interfaces>
      <Interface name=("name")>
        <IPAddress>(IPv6 address)</IPAddress>
        <Mask>255.255.255.0</Mask>
        <IPv6Address>(IPv6 address)/(Prefix Length)</IPv6Address>
        <NdpAdvSendAdvertisements>(yes/no)</NdpAdvSendAdvertisements>
        <NdpAdvPrefix>(IPv6 prefix)/(Prefix Length)</NdpAdvPrefix>
        <NdpMaxRtrAdvInterval>(4-1800)</NdpMaxRtrAdvInterval>
        <NdpMinRtrAdvInterval>(3-1350)</NdpMinRtrAdvInterval>
        <Bandwidth>1000</Bandwidth>
        <Duplex>auto</Duplex>
        <Speed>auto</Speed>
        <shutdown></shutdown>
      </Interface>
    </Interfaces>
    <Routing6>
      <Static>
        <Route>
          <NetworkAddress>(IPv6 prefix)/(Prefix Length)</NetworkAddress>
          <NextHopAddress>(IPv6 address)</NextHopAddress>
          <AdministrativeDistance>(1-254)</AdministrativeDistance>
        </Route>
      </Static>
    </Routing6>
  </Router>
</Routers>
```

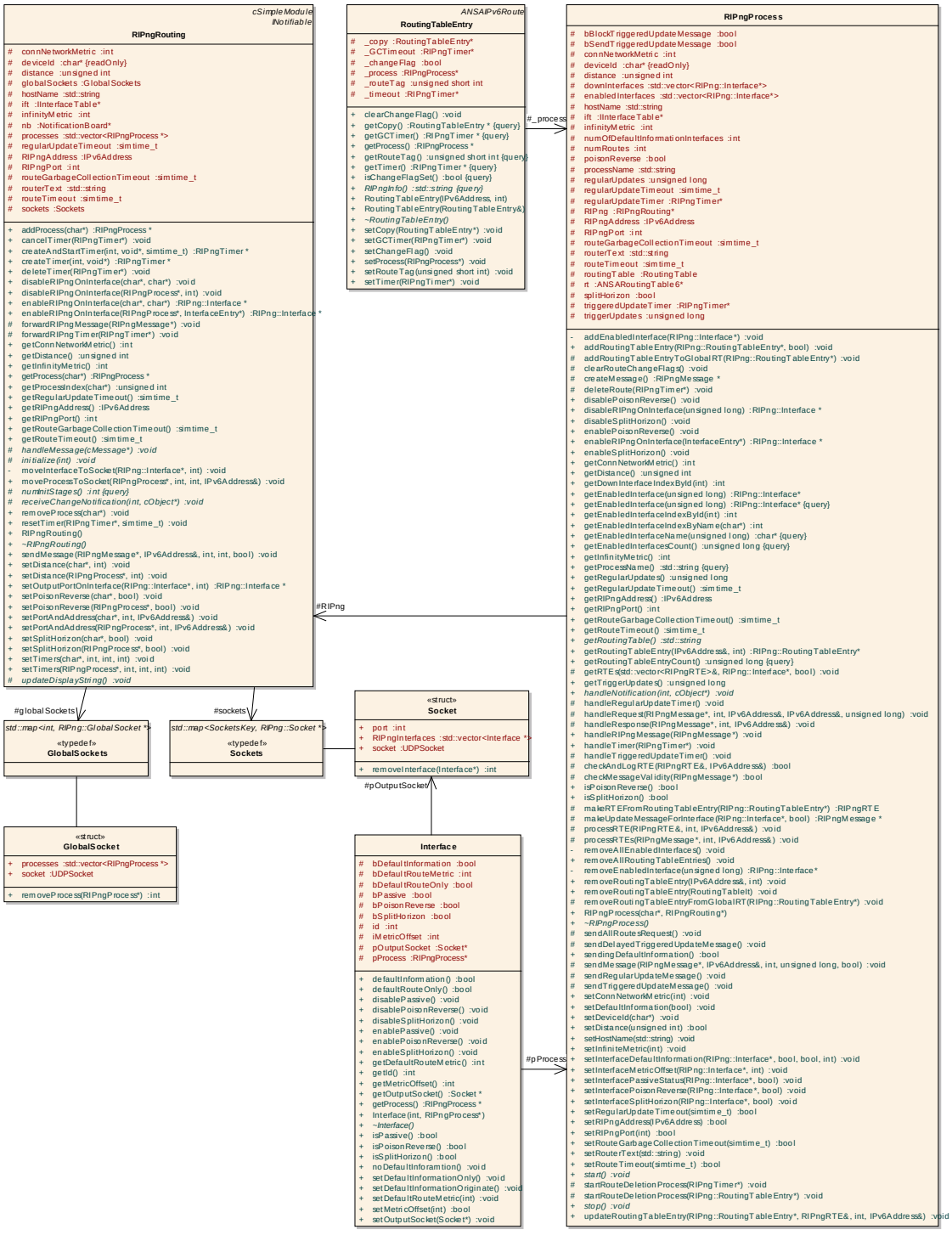
Příloha B

Seznam souborů a diagram tříd

Následující seznam uvádí soubory (a jejich umístění v projektu ANSA), které byly vytvořeny nebo upraveny v rámci této práce. V současné době jsou zdrojové kódy projektu ANSA dostupné ve veřejném repozitáři https://github.com/kvetak/ANSA.git.

```
src
└── ansa
    ├── applications
    │   └── ripng
    │       ├── RIPngInterface.cc
    │       ├── RIPngInterface.h
    │       ├── RIPngProcess.cc
    │       ├── RIPngProcess.h
    │       ├── RIPngRouting.cc
    │       ├── RIPngRouting.h
    │       ├── RIPngRoutingTableEntry.cc
    │       ├── RIPngRoutingTableEntry.h
    │       ├── RIPngMessage.msg
    │       ├── RIPngRouter.ned
    │       ├── RIPngRouting.ned
    │       └── RIPngTimer.msg
    ├── networklayer
    │   └── ipv6
    │       ├── ANSARoutingTable6.cc
    │       ├── ANSARoutingTable6.h
    │       ├── ANSARoutingTable6Access.h
    │       └── ANSARoutingTable6.ned
    ├── nodes
    │   └── ipv6
    │       └── ANSAStandardHost6.ned
    └── transport
        └── udp
            ├── ANSAUDP.cc
            ├── ANSAUDP.h
            └── ANSAUDP.ned
```

```
src
└── ansa
    └── util
        └── deviceConfigurator
            ├── deviceConfigurator.cc
            ├── deviceConfigurator.h
            ├── xmlParser.cc
            └── xmlParser.h
```



Obrázek B.1: Diagram tříd RIPng modulu.

Příloha C

Konfigurační soubory scénářů

Uvedené konfigurační soubory byly použity při ověření funkčnosti implementace (popsáno v kapitole 9). Konfigurace `config.xml` je výchozí. V ostatních konfiguračních souborech jsou uvedena pouze zařízení s rozdílným nastavením oproti `config.xml`.

C.1 `RIPngTest1 (config.xml)`

```
<Devices>
    <Router id="2001:12::1">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>12.12.12.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:12::1/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
            <Interface name="eth2">
                <IPAddress>1.1.1.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:1:1:1::1/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>2.2.2.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:2:2:2::2/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>13.13.13.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:13::1/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
        </Interfaces>
```

```
    <Routing6>
     <RIPng name="RIPng1"/>
    </Routing6>
</Router>

<Router id="2001:12::2">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>12.12.12.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:12::2/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
        <Interface name="eth2">
            <IPAddress>3.3.3.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:3:3:3::3/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth3">
            <IPAddress>4.4.4.4</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:4:4:4::4/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth1">
            <IPAddress>23.23.23.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:23::2/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
    </Interfaces>

    <Routing6>
     <RIPng name="RIPng1"/>
    </Routing6>
</Router>

<Router id="2001:13::3">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>13.13.13.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:13::3/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
        <Interface name="eth2">
            <IPAddress>5.5.5.5</IPAddress>
            <Mask>255.255.255.0</Mask>
```

```
                <IPv6Address>2001:5:5:5::5/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>6.6.6.6</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:6:6:6::6/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>23.23.23.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:23::3/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
            <Interface name="eth4">
                <IPAddress>34.34.34.3</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:34::3/64</IPv6Address>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>

    <Router id="2001:34::4">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>34.34.34.4</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:34::4/64</IPv6Address>
            </Interface>
            <Interface name="eth1">
                <IPAddress>7.7.7.7</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:7:7:7::7/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>
</Devices>
```

C.2 RIPngTest2 (config2.xml)

```
<Devices>
    <Router id="2001:13::3">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>13.13.13.3</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:13::3/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
            <Interface name="eth2">
                <IPAddress>5.5.5.5</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:5:5:5::5/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>6.6.6.6</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:6:6:6::6/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>23.23.23.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:23::3/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
            <Interface name="eth4">
                <IPAddress>34.34.34.3</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:34::3/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>

    <Router id="2001:34::4">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>34.34.34.4</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:34::4/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
```

```
            <Interface name="eth1">
                <IPAddress>7.7.7.7</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:7:7:7::7/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>
</Devices>
```

C.3 RIPngTest3 (config3.xml)

```
<Devices>
    <Router id="2001:12::1">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>12.12.12.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:12::1/64</IPv6Address>
            </Interface>
            <Interface name="eth2">
                <IPAddress>1.1.1.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:1:1:1::1/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>2.2.2.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:2:2:2::2/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>13.13.13.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:13::1/64</IPv6Address>
                <RIPng name="RIPng1"/>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>
```

```
<Router id="2001:12::2">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>12.12.12.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:12::2/64</IPv6Address>
        </Interface>
        <Interface name="eth2">
            <IPAddress>3.3.3.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:3:3:3::3/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth3">
            <IPAddress>4.4.4.4</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:4:4:4::4/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth1">
            <IPAddress>23.23.23.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:23::2/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
    </Interfaces>

    <Routing6>
      <RIPng name="RIPng1">
         <Address>FF02::99</Address>
         <Port>527</Port>
       </RIPng>
    </Routing6>
</Router>

<Router id="2001:13::3">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>13.13.13.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:13::3/64</IPv6Address>
            <RIPng name="RIPng1">
             <DefaultInformation>
             <Metric>5</Metric>
             <DefaultOnly>true</DefaultOnly>
             </DefaultInformation>
            </RIPng>
        </Interface>
        <Interface name="eth2">
```

```
            <IPAddress>5.5.5.5</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:5:5:5::5/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth3">
            <IPAddress>6.6.6.6</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:6:6:6::6/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth1">
            <IPAddress>23.23.23.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:23::3/64</IPv6Address>
            <RIPng name="RIPng2">
             <DefaultInformation/>
            </RIPng>
        </Interface>
        <Interface name="eth4">
            <IPAddress>34.34.34.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:34::3/64</IPv6Address>
            <RIPng name="RIPng1">
             <DefaultInformation/>
            </RIPng>
        </Interface>
    </Interfaces>

    <Routing6>
     <RIPng name="RIPng1"/>
     <RIPng name="RIPng2">
        <Address>FF02::99</Address>
        <Port>527</Port>
     </RIPng>
    </Routing6>
</Router>

<Router id="2001:34::4">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>34.34.34.4</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:34::4/64</IPv6Address>
            <RIPng name="RIPng1">
             <DefaultInformation/>
            </RIPng>
        </Interface>
        <Interface name="eth1">
            <IPAddress>7.7.7.7</IPAddress>
```

```
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:7:7:7::7/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>
</Devices>
```

C.4 RIPngTest4 (config4.xml)

```
<Devices>
    <Router id="2001:12::1">
        <Interfaces>
            <Interface name="eth0">
                <IPAddress>12.12.12.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:12::1/64</IPv6Address>
                <RIPng name="RIPng1">
                 <SplitHorizon>disable</SplitHorizon>
                </RIPng>
            </Interface>
            <Interface name="eth2">
                <IPAddress>1.1.1.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:1:1:1::1/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>2.2.2.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:2:2:2::2/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>13.13.13.1</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:13::1/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PoisonReverse>enable</PoisonReverse>
                </RIPng>
            </Interface>
        </Interfaces>

        <Routing6>
```

```
     <RIPng name="RIPng1"/>
    </Routing6>
</Router>

<Router id="2001:12::2">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>12.12.12.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:12::2/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
        <Interface name="eth2">
            <IPAddress>3.3.3.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:3:3:3::3/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth3">
            <IPAddress>4.4.4.4</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:4:4:4::4/64</IPv6Address>
            <RIPng name="RIPng1">
             <PassiveInterface>enable</PassiveInterface>
            </RIPng>
        </Interface>
        <Interface name="eth1">
            <IPAddress>23.23.23.2</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:23::2/64</IPv6Address>
            <RIPng name="RIPng1"/>
        </Interface>
    </Interfaces>

    <Routing6>
     <RIPng name="RIPng1">
     <Timers>
     <Update>10</Update>
     <Route>3</Route>
     <Garbage>2</Garbage>
     </Timers>
     </RIPng>
    </Routing6>
</Router>

<Router id="2001:13::3">
    <Interfaces>
        <Interface name="eth0">
            <IPAddress>13.13.13.3</IPAddress>
            <Mask>255.255.255.0</Mask>
            <IPv6Address>2001:13::3/64</IPv6Address>
            <RIPng name="RIPng1"/>
```

```
            </Interface>
            <Interface name="eth2">
                <IPAddress>5.5.5.5</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:5:5:5::5/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth3">
                <IPAddress>6.6.6.6</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:6:6:6::6/64</IPv6Address>
                <RIPng name="RIPng1">
                 <PassiveInterface>enable</PassiveInterface>
                </RIPng>
            </Interface>
            <Interface name="eth1">
                <IPAddress>23.23.23.2</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:23::3/64</IPv6Address>
                <RIPng name="RIPng1">
                 <MetricOffset>3</MetricOffset>
                </RIPng>
            </Interface>
            <Interface name="eth4">
                <IPAddress>34.34.34.3</IPAddress>
                <Mask>255.255.255.0</Mask>
                <IPv6Address>2001:34::3/64</IPv6Address>
            </Interface>
        </Interfaces>

        <Routing6>
         <RIPng name="RIPng1"/>
        </Routing6>
    </Router>
</Devices>
```