VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

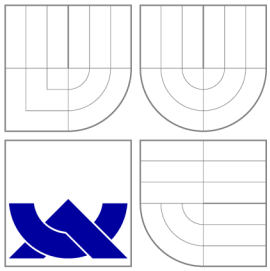
MODELOVÁNÍ MULTICASTOVÉHO SMĚROVÁNÍ V PROSTŘEDÍ OMNET++

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

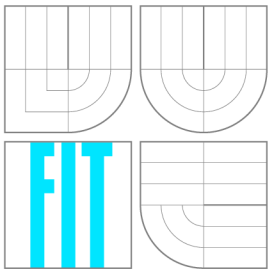
AUTOR PRÁCE
AUTHOR

Bc. VERONIKA RYBOVÁ

BRNO 2012



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ MULTICASTOVÉHO SMĚROVÁNÍ V PROSTŘEDÍ OMNET++

MULTICAST ROUTING MODELLING IN OMNET++

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. VERONIKA RYBOVÁ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2012

Abstrakt

V dnešních sítích se běžně setkáváme s multicastovým provozem. Pro seznámení se s multicastovou architekturou a jejím chováním v jakékoliv situaci je nejlepší využít možnosti simulování. Tato diplomová práce se zabývá modelováním a simulováním multicastového směrování v nástroji OMNeT++. Text seznámí čtenáře s protokolem PIM a jeho jednotlivými módy (DM, SM, SSM a BiDir) s důrazem na PIM-DM. Práce se především zaměřuje na návrh a implementaci rozšíření nástroje OMNeT++ o multicastové směrování protokolem PIM-DM. Správnost implementace je ověřena porovnáním simulace a reálné sítě na příkladu.

Abstract

Multicast traffic is common in today's networks. We need to simulate multicast architecture to be familiarized with its functionality in all situations. This thesis describes modelling and simulation of multicast routing using OMNeT++ tool. The text introduces protocol PIM and its particular modes (DM, SM, SSM, and BiDir) with emphasis on PIM-DM. The thesis focuses especially on design and implementation of OMNeT++ extension by multicast routing protocol PIM-DM. Correctness of implementation is verified by comparison of simulation and real network on example.

Klíčová slova

Simulace sítí, modelování sítí, multicast, multicastové směrování, PIM, PIM-DM, PIM-SM, OMNeT++, INET, Cisco, ANSA

Keywords

Simulation of networks, modelling of networks, multicast, multicast routing, PIM, PIM-DM, PIM-SM, OMNeT++, INET, Cisco, ANSA

Citace

Veronika Rybová: Modelování multicastového směrování
v prostředí OMNeT++, diplomová práce, Brno, FIT VUT v Brně, 2012

Modelování multicastového směrování v prostředí OMNeT++

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Vladimíra Veselého

.....
Veronika Rybová
22. května 2012

Poděkování

Na tomto místě bych ráda v prvé řadě poděkovala Vláďovi za jeho pomoc při řešení diplomové práce, za jeho neutuchající humor, který člověku vždy pozvedne náladu, i za informace a zkušenosti, které mi předával rok co rok v rámci Cisco akademie. Děkuji všem svým kamarádům-spolužákům, kteří mi pomáhali udržovat si zdravý rozum i v těch nejkritičtějších fázích studia na FIT.

Největší poděkování si ale zaslouží moji rodiče, bez kterých bych teď nebyla tam, kde jsem. Děkuji vám za nemalou finanční podporu, kterou moje studium vyžadovalo, za úsilí, které vás stála moje výchova, a především za vaši psychickou podporu, která mi vždycky pomohla v životě překonat všechny překážky. Na závěr bych také ráda poděkovala svému příteli za to, že mě vždy podporuje v tom, co dělám.

Obsah

Seznam obrázků

Úvod

V dnešní době se multicastový provoz využívá čím dál více. Setkáme se s ním především při streamingu médií jako je internetová televize či rádio. Streamovat je možné i videokonference nebo přednášky z vysokých škol či podobných institucí. Ve firemních sítích se multicast používá také například pro rozesílání aktualizací nebo pro sledování lokálních videokonferencí. Oproti unicastu a broadcastu využívá multicast mnohem lépe šířku pásma a to je důvod, proč se rozšiřuje jeho pole působnosti.

Unicast posílá data pouze z jednoho zdroje do jednoho cíle. Pokud chceme odeslat data více uživatelům, bylo by nutné je zasílat na několikrát, čímž bychom zbytečně plýtvali šířkou pásma. Broadcast zasílá data všem uživatelům v síti bez ohledu na to, zda o ně mají či nemají zájem. V případě, že o ně má zájem jen část uživatelů také dochází k plýtvání šířky pásma a navíc k zbytečné zátěži koncových stanic, které data musí zpracovat, i když o ně nemají zájem.

Tyto problémy daly vzniknout právě multicastu. Je vhodný tam, kde o data má zájem více stanic, ale ne všechny. Zdroj vysílá vždy jen jednu kopii dat. Na směrovačích v síti je, aby případně data nakopírovaly a rozeslaly do sítí, kde se nachází cílové stanice.

Tato práce se zabývá tématikou směrování multicastového provozu. Ten je klíčový pro správné fungování multicastu a je značně odlišný od směrování unicastového. Z tohoto důvodu vznikly multicastové směrovací protokoly, které dokáží zajistit propojení mezi zdroji a často se měnícími cíli multicastového provozu.

Rozhodneme-li se nasadit multicast na nějaké rozsáhlejší síti, jistě není vhodné experimentovat na reálné topologii. Nasazení vyžaduje i testování různého nastavení a parametrů multicastu. Tyto testy v reálné síti by pravděpodobně měly za následek nespokojené uživatele, ztráty dat a přinejhorším zcela nefunkční síť.

Jako nejlevnějším a nejefektivnějším řešení se jeví využití simulace sítě. V simulačním nástroji můžeme modelovat aktuální nebo zamýšlenou topologii sítě a podrobit ji sérii nejrůznějších testů. Dnešní simulační nástroje většinou podporují i grafické uživatelské rozhraní, díky kterému je možné vizuálně sledovat chování sítě.

Dobrým simulačním nástrojem je OMNeT++. Ten je ale stále ve vývoji a i když zvládá simulaci čím dál většího množství běžně využívaných protokolů, jeho součástí není žádný multicastový směrovací protokol, ani větší podpora multicastu.

Cíl práce

Tato diplomová práce vznikla v rámci fakultního projektu ANSA, který si klade za cíl mimo jiné využít nástroj OMNeT++ k simulaci VUT sítě. Aby bylo možné cíl realizovat, je nutné doplnit funkcionalitu nástroje tak, aby byl schopen simulovat běžný provoz školní sítě. Jeho nedílnou součástí je také provoz multicastový (např. streaming přednášek).

Začneme analýzou možností simulačního nástroje OMNeT++ na poli multicastu. Vybereme vhodný multicastový protokol z rodiny PIM protokolů, ten naimplementujeme a integrujeme do nástroje. Následně provedeme jeho otestování a srovnání s reálnou síti. Využijeme sítě s Cisco prvky, které jsou na škole dostupné a zároveň patří mezi špičku toho, co se v korporátních sítích dnes používá.

Aby text nebyl přespříliš rozsáhlý, nezabíhali jsme do přílišných detailů a mohli jsme se zaměřit na náš cíl, předpokládáme, že čtenář má základní znalosti o fungování sítí, multicastu a směrování v sítích.

Struktura práce

První kapitola nás seznámí se simulačním nástrojem OMNeT++, knihovnou INET a ANSA-INET. Popisuje také současné možnosti multicastu v prostředí OMNeT++ a INET.

V druhé kapitole se podíváme na fungování multicastových směrovacích protokolů se zaměřením na protokoly rodiny PIM. Shrneme si nejdůležitější vlastnosti protokolů PIM-SM, PIM-SSM, BiDir-PIM a podrobněji si rozebereme protokol PIM-DM.

Třetí kapitola nám přiblíží, jak je možné konfigurovat směrovací protokoly PIM na reálných zařízeních. Zaměřili jsme se na směrovače značky Cisco.

Čtvrtá kapitola se zabývá návrhem rozšíření simulátoru o podporu multicastového směrování, konkrétně o protokol PIM-DM. Zmíněn je jak návrh struktury, abstraktních datových typů a procesů.

Pátá kapitola zahrnuje vše, co se týká konfigurace zařízení v simulátoru. Vytvoříme si strukturu konfiguračního souboru a popíšeme si související implementační kroky.

Samotná implementace je podrobně probrána v kapitole šest. Zabývá se nejen implementací protokolu PIM-DM, ale také implementací podpůrných prostředků jako je multicastová směrovací tabulka, tabulka rozhraní nebo tabulka sousednosti.

V závěrečné kapitole sedm provedeme porovnání simulace s reálnou sítí v prostředí Cisco zařízení s cílem dokázat správnost naší implementace.

Kapitola 1

Simulační nástroj OMNeT++ a knihovna INET

Začneme seznámením se s nástrojem OMNeT++ a knihovnou INET, které budeme v rámci této práce používat. Prozkoumáme možnosti tohoto nástroje a existující podporu pro simulaci multicastového provozu.

1.1 OMNeT++

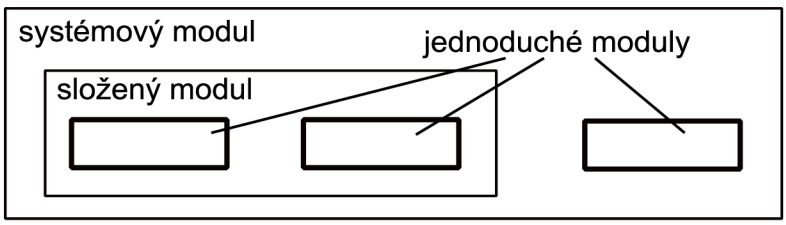
OMNeT++ je pro akademické účely volně dostupný simulační nástroj s otevřenými zdrojovými kódy. Pro popis jsme zvolili jeho verzi 4.1. Všechny informace byly čerpány z velmi dobře zpracovaného manuálu [21].

1.1.1 Možnosti programu OMNeT++

OMNeT++ je objektově orientovaný modulární diskrétní síťový simulátor. Je založený na objektovém jazyku C++ a vlastním jazyku NED. Existují verze pro Unix i Windows a obě verze jsou pro školní účely zdarma. Kromě simulačního jádra obsahuje GUI a IDE. Obsahuje knihovny pro práci s TCP/IP, Ethernet, FDDI, Token Ring, 802.11 a Peer-to-peer.

1.1.2 Modelování

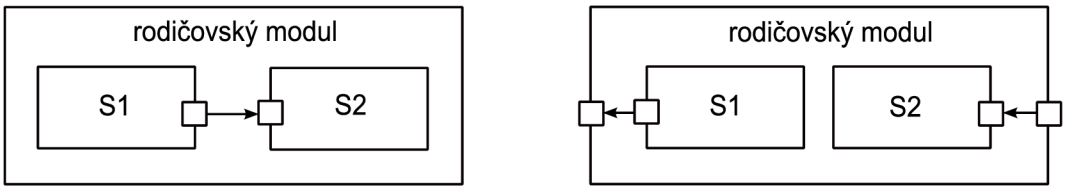
Při modelování v OMNeT++ se do sebe vnořují jednotlivé moduly hierarchicky. Nejvýše postavený modul se označuje jako modul systémový, ten se skládá ze submodulů (obrázek 1.1). Může se jednat o modely složené, které se skládají z dalších složených modulů nebo z modulů jednoduchých. Ty jsou dále nedělitelné.



Obrázek 1.1: Hierarchická struktura modulů.

Topologie sítě – propojení jednotlivých modulů – se v OMNeT++ popisuje speciálním jazykem NED. Jednoduché moduly obsahují algoritmus, který se zapisuje v jazyce C++. Moduly není třeba psát od začátku. OMNeT++ obsahuje několik předdefinovaných tříd objektů, u kterých je třeba jen redefinovat chování.

Moduly mezi sebou komunikují pomoci zasílání zpráv. Ty budeme většinou považovat za model PDU. Zprávy mohou přijít od jiného modulu nebo ze stejného (využívají se jako časovače). Každý modul obsahuje brány, které jsou vstupní (In) pro příjem zpráv a výstupní (Out) pro odesílání zpráv. Mezi bránami modulů se vytváří spojení. Spojení může existovat mezi moduly na stejné úrovni hierarchie nebo mezi modulem a jeho složeným rodičovským modulem (obrázek 1.2). Pro možnost modelování přenosu paketů po lince, má spojení tři volitelné parametry – přenosové zpoždění, bitová chybovost a rychlost přenosu dat.



Obrázek 1.2: Spojení submodulů mezi sebou a propojení rodičovského modulu se submoduly.

Jazyk NED a způsob simulace v OMNeT++ je blíže popsán v [16].

1.2 INET

Framework INET je volně šířená knihovna pro simulační prostředí OMNeT++. Rozšiřuje tak základní schopnosti OMNeT++ o některé drátové i bezdrátové síťové protokoly. Mimo jiné obsahuje protokoly UDP, TCP, SCTP, IP, IPv6, Ethernet, PPP, 802.11, MPLS, OSPF a další. [2]

V rámci projektu ANSA[1] dochází k rozšiřování této knihovny, které se označuje jako ANSAINET. Nad rámec knihovny INET obsahuje moduly pro protokoly RIP, OSPF a IGMP, modul pro ACL filtrování, schopnost načítat konfiguraci sítě z konfiguračních souborů, model přepínače, podpora VLAN a protokolu STP, podpora dual-stack[1] architektury směrovače i hosta, protokol OSPFv3, atd. Programový výstup této práce bude také integrován do knihovny ANSAINET.

1.3 Současné možnosti multicastu v prostředí OMNeT++ a INET

Samotný OMNeT++ nenabízí žádnou podporu multicastu. Ani knihovna INET na tom není lépe. Na podpoře multicastu se začalo prakticky pracovat až v rámci projektu ANSA.

1.3.1 Podpora v knihovně INET

Po instalaci OMNeT++ a importování knihovny INET není možné provádět žádné simulace reálného multicastového provozu. Autoři ale počítali s budoucí implementací multicastu,

[1] Duální implementace, zařízení podporuje IPv4 i IPv6

a tak je možné v jeho síťové vrstvě (INET/network layer/ipv4) nalézt v některých modulech základní podporu.

Modul `IP`, který implementuje IPv4 protokol, obsahuje funkci `routeMulticastPacket()`. Ta rozliší, zda paket přichází ze sítě, či je zdrojem samotný směrovač. Následuje vlastní směrování, při kterém načte cesty ze směrovací tabulky. Pro každou cestu nakopíruje paket a rozešle.

Modul `Routing Table`, který implementuje směrovací tabulku, obsahuje strukturu multicastové cesty (`MulticastRoute`). Ta se ale vůbec nepodobá reálnému zápisu multicastové cesty. Jako použitelná se jeví pouze pomocná funkce `isLocalMulticastAddress()`, která kontroluje, jestli je adresa na seznamu lokálních multicastových skupin.

1.3.2 Podpora v knihovně ANSAINET

Lepší podporu lze nalézt v knihovně ANSAINET, která vznikla v rámci projektu ANSA. Zásadní je přidání modulu, který implementuje protokol IGMPv2. Tento protokol se využívá pro dynamické přihlašování a odhlašování ze skupiny u multicastového směrovače ve své lokální síti [24][12].

Modul [14] obsahuje kromě vlastní protokolové funkcionality IGMP tabulku rozhraní a tabulku multicastových skupin. Zásadní bude funkce `processMembershipReportV2()`, ve které by mělo dojít k propojení s multicastovým směrováním. Zřejmě bude také nutné upravit funkci `isRouter()`, která rozpoznává směrovač podle jeho názvu, přičemž předpokládá, že jeho název bude obsahovat řetězce "router"nebo "Router".

1.4 Shrnutí

V této kapitole jsme se seznámili se simulačním prostředím OMNeT++ a knihovnami INET a ANSAINET, které budeme používat k další práci. Knihovna INET má velmi slabou podporu multicastu. Důležitá je implementace protokolu IGMP v knihovně ANSAINET. Simulační prostředí ale nemá žádné prostředky pro dynamické multicastové směrování. To bude třeba v rámci této práce naimplementovat spolu s některými dalšími podpůrnými prostředky.

Kapitola 2

Multicastové směrování v sítích a Protokol Independent Multicast (PIM)

V této kapitole se seznámíme se základními poznatky o multicastovém směrování. Podrobně se podíváme na protokol PIM-DM, konkrétně nás bude zajímat, čím je řízena jeho logika a jaké zprávy používá. Zběžně si představíme i protokoly PIM-SM, PIM-SSM a BiDir-PIM.

2.1 Multicastové směrování v sítích

Uživatel se může připojit do skupiny a přijímat data zasílaná do multicastové skupiny. Nyní je třeba vyřešit, jak se data dostanou od zdroje k uživateli. Toto zajišťují multicastové směrovací protokoly.

2.1.1 Multicastové stromy

Než se dostaneme k samotnému směrování v multicastu, je nutné si vysvětlit některé pojmy, se kterými pracují. Základem pro tyto směrovací protokoly jsou distribuční stromy. Multicastová data jsou pak směrovány právě podle takovýchto stromů. Existují dva základní typy:

- Zdrojový strom (Source Tree) - Pro každý zdroj (kořen) v multicastové skupině se vybuduje strom nejkratších vzdáleností ke všem příjemcům (listy). Pokud je ale v síti velké množství zdrojů či příjemců, udržování zdrojových stromů je velmi náročné na výpočetní prostředky směrovačů.
- Sdílený strom (Shared Tree) - V síti existuje zařízení (Randevous Point, RP) sdružující provoz všech zdrojů multicastu, strom se pak buduje od RP (kořen) ke všem příjemcům (listy). Sdílený strom šetří výpočetní prostředky, na druhou stranu, cesta od zdroje k cíli nemusí být vždy optimální.

Zdrojový strom se označuje jako dvojice (S,G), kde S je adresa zdroje multicastového provozu a G je adresa multicastové skupiny. Sdílený strom se označuje jako (*,G).

2.1.2 PIM

Některé Multicastové směrovací protokoly se souhrnně označují zkratkou PIM (Protocol Independent Multicast). Označují se jako protokolově nezávislé proto, že sami o sobě nezjišťují topologii sítě, ale využívají informace z unicastových směrovacích protokolů.

Základní rozdíl mezi unicastovým a multicastovým směrováním je v tom, že unicastové směrování je založeno na cílové adrese paketu, zatímco multicastové směrování je založeno na zdrojové IP adrese.

Exitují 4 základní PIM módy:

- PIM-DM (PIM Dense Mode)
- PIM-SM (PIM Sparse Mode, v dnešní době nejpoužívanější)
- BiDir-PIM (Bidirectional PIM)
- PIM-SSM (PIM Source-Specific Multicast)

Pro zdroj neexistuje žádný protokol pro komunikaci, registraci či informování směrovačů. Zdroj jednoduše začne vysílat do sítě data, na sousedních směrovačích pak je, aby je dále rozdistribuovaly ke svým sousedům. Nikdy však data neposílají zpět směrem k zdroji, což zabraňuje vzniku smyček. Pro konkrétní implementaci se využívá technika RPF (Reverse path forwarding)[25].

Jakmile je multicastový paket přijat na některém rozhraní, provede se RPF kontrola. Směrovač porovná zdrojovou IP adresu paketu s unicastovou směrovací tabulkou. Pokud souhlasí zdrojová síť s rozhraním, na které paket přišel (RPF rozhraní), RPF kontrola proběhla v pořádku. Směrovač následně rozešle paket na všechny ostatní rozhraní, která jsou součástí dané multicastové skupiny. Pokud RPF kontrola neprojde, paket je zahozen. Kontroluje se reverzní cesta, proto se tento mechanismus nazývá RPF.

Proto, aby protokol fungoval správně, je nutné, aby směrovací tabulka byla bezchybná a již zkonvergovaná. Dalším podstatným předpokladem pro fungovaní RPF je, že cesty od zdroje k cíli a naopak jsou symetrické. Pokud jsou asymetrické, RPF kontrola selže.

2.2 Protocol Independent Multicast - Dense Mode (PIM-DM)

PIM-DM [3] je multicastový směrovací protokol, který využívá unicastové směrovací informace jako základ pro šíření multicastových datagramů na všechny multicastové směrovače. Směrovače, které nejsou součástí multicastové skupiny a nemají tudíž o zprávy zájem, na to explicitně upozorňují zdroj zasláním speciálních zpráv. Je to zásadní rozdíl oproti PIM-SM, který zasílá multicast jen, když je vyžadován.

2.2.1 Základní informace o PIM-DM

Na začátku multicast od zdroje zaplaví celou síť. Všechny směrovače, jejichž lokální hosté nejsou součástí multicastové skupiny, se odříznou od zdrojového stromu (Prune Off) a přejdou do stavu Prune (asociovaný s určitým (S,G)). Tento stav má svůj časovač, po jeho vypršení se do odříznuté větve začne znovu zasílat multicast. V případě, že se ve skupině G objeví nový příjemce, směrovač se může ke zdrojovému stromu znovu připojit.

V síti se v pravidelných intervalech střídají dva cykly - Flood (zaplavení) a Prune (odřezávání). To může vést na velkou zátěž sítě. Opakovaným záplavám ale lze zabránit pomocí State Refresh zprávy zasílané směrovači přímo připojenými na zdroj. Zpráva se šíří sítí, a jakmile je přijata směrovačem na RPF rozhraní dojde k obnovení časovače stavu Prune - větev zůstává i nadále odříznuta.

Zásadní rozdíl mezi PIM-SM a PIM-DM je ten, že PIM-SM zasílá data na směrovač jen tehdy, pokud je někdo vyžaduje, což šetří šířku pásma, výpočetní prostředky směrovačů atd. Oproti tomu PIM-DM nepotřebuje RP, což je podstatné pro sítě, které nemohou tolerovat jeden kritický bod v síti (single point of failure). Přestane-li fungovat RP, přestane fungovat multicastové směrování.

Informace z unicastové směrovací tabulky se využijí pro vybudování multicastové směrovací informační báze (Multicast Routing Information Base (MRIB)) a její údržbu. MRIB slouží především k tomu, aby poskytovala adresu next-hop směrovače (následující skok) pro zasílání PIM Join/Prune zpráv ve stromě směrem ke zdroji. Data pak putují přesně opačným směrem (reverzní cestou). Všimněme si, že unicastová směrovací tabulka funguje zcela obráceně - poskytuje next-hop ve směru k příjemci, ne zdroji.

Každý směrovač má Tree Information Base (TIB). Tato databáze obsahuje množinu stavů pro všechny distribuční stromy uložené v směrovači. Stavy se mění na základě PIM zpráv a IGMP zpráv od lokálních hostů.

TIB je zásadní pro zasílání multicastových datagramů, což uvidíme i v sekci 2.2.3. Většina implementací ale nespoléhá pouze na TIB, neboť její používání při směrování je poněkud neohrabané. V praxi se spíše používá Multicast Forwarding Information Base (MFIB), která se sestaví na základě stavů uložených v TIB tabulce.

Informace (i stavové) pro zdrojový strom (S,G) se uchovávají jen po dobu, kdy je se stromem spjatý nějaký aktivní časovač. V momentě, kdy není žádný časovač aktivní, informace o stromu jsou ze směrovače vymazány.

Směrovač uchovává tyto stavy a časovače pro každé rozhraní:

- Hello Timer (HT)
- State Refresh Capable
- LAN Delay Enabled
- Propagation Delay (PD)
- Override Interval (OI)
- Neighbor State (pro každého souseda):
  - Informace ze zprávy Hello od souseda
  - Neighbor's Gen ID.
  - Neighbor's LAN Prune Delay
  - Neighbor's Override Interval
  - Neighbor's State Refresh Capability
  - Neighbor Liveness Timer (NLT)

Pro každou dvojici (S,G) směrovač uchovává tyto stavy a časovače:

- Pro každé rozhraní:

  - Local Membership (NoInfo nebo Include)
  - (S,G) Prune State (NoInfo, Pruned nebo PrunePending) + časovače: Prune Pending Timer (PPT), Prune Timer (PT)
  - (S,G) Assert Winner State (NoInfo, Loser, Winner) + časovač Assert Timer (AT), IP adresa vítěze a jeho metrika

- Pro upstream[1] rozhraní:
  - Graft/Prune State (Pruned, Forwarding nebo AckPending) + časovače: Graft Retry Timer (GRT), Override Timer (OT), Prune Limit Timer (PLT)
  - Originator State (Originator nebo NotOriginator) + časovače: Source Active Timer (SAT), State Refresh Timer (SRT)

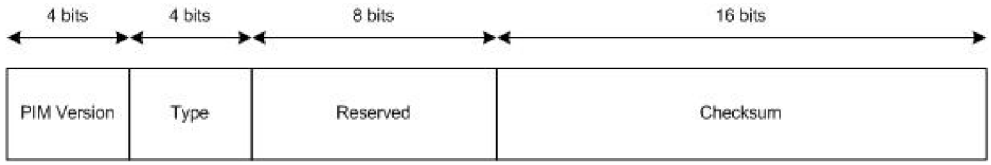
Pro každý přijatý paket se provede stejná procedura. Nejprve se zkontroluje, zda byla zpráva přijata na PRF rozhraní pro danou skupinu (případně se provede RPF kontrola). Pokud ano, vytvoří se seznam rozhraní (oilist), na které se má zpráva přeposlat. Pokud je seznam prázdný, zpráva se zahodí a zahájí se proces odříznutí od stromu, jinak se zpráva přepošle na všechny rozhraní seznamu.

2.2.2 PIM-DM zprávy

PIM-DM používá tyto zprávy:

- Hello Message
- PIM-DM Prune Message
- PIM-DM Join Message
- PIM-DM Graft Message
- State Refresh
- PIM Assert Message

Všechny zprávy mají stejný formát jako protokol PIM-SM, proto úplnou specifikaci zpráv najdeme v [11]. IP protokol má číslo 103 a TTL musí být nastaveno na 1. Všechny PIM-DM zprávy kromě Graft jsou vždy zasílány na multicastovou adresu 224.0.0.13 (ff02::d u IPv6). Graft zpráva se zasílá jako jediná unicastem.



Obrázek 2.1: Formát PIM hlavičky.

Hlavička (obr. 2.1) je pro všechny typy zpráv shodná. Verze protokolu PIM je nastavena na 2, pole reserved se nastavuje na 0, checksum obsahuje standartní IP kontrolní součet.

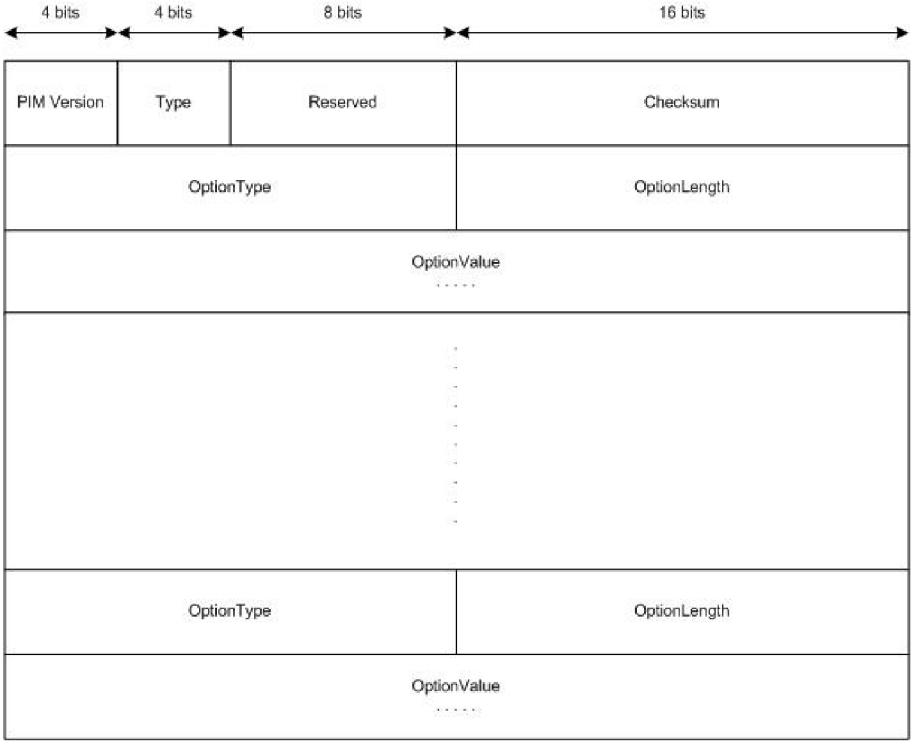
[1] Upstream rozhraní je rozhraní ve směru ke zdroji multicastu. Označuje se také jako RPF rozhraní.

Type označuje typ PIM zprávy: 0 = Hello; 1 = Register (pouze PIM-SM); 2 = Register Stop (pouze PIM-SM); 3 = Join/Prune; 4 = Bootstrap (pouze PIM-SM); 5 = Assert; 6 = Graft; 7 = Graft Ack; 8 = Candidate RP Advertisement (pouze PIM-SM); 9 = State Refresh.

PIM Hello zpráva

Pro detekci sousedů a navázání spojení s nimi se používá PIM Hello zpráva. Ta se periodicky posílá multicastem na všechny rozhraní směrovače, na kterých je povolen multicast. Jakmile povolíme multicast na rozhraní nebo se směrovač nastartuje, nastaví se Hello Timer na náhodnou hodnotu mezi 0 a 5 sekundami. Po této inicializaci se zpráva zasílá každých 30 sekund. Používá se jeden časovač pro všechna rozhraní.

Důležité také je, i z bezpečnostního hlediska, že žádný směrovač nepřijme od svého souseda žádnou PIM zprávu, pokud od něj před tím neobdržel Hello zprávu. Pakliže nějaký směrovač potřebuje zaslat svému sousedovi PIM Join/Prune/Assert zprávu, musí bez odkladu (bez nastavení náhodného intervalu) poslat Hello zprávu.



Obrázek 2.2: Formát PIM Hello zprávy.

Tělo PIM Hello zprávy (obrázek 2.2) používá kódování TLV (Type-Length-Value)[26]. Každá položka se skládá ze tří bloků typ (Option Type), délka (Option Length) a hodnota (Option Value). Zpráva se může skládat z několika takových položek. Používají se tyto typy:

1 = Hello Hold Time; 2 = LAN Prune Delay; 19 = DR Priority (pouze PIM-SM); 20 = Generation ID; 21 = State Refresh Capable; 22 = Bidir Capable.

Hello Hold Time je počet sekund, po které musí příjemce udržovat informace o Hello zprávě (rozhraní, odesílatel, informace ze zprávy). LAN Prune Delay slouží pro ustavení

některých časovačů, které zabraňují záplavě směrovače zprávami Join, po té, co jeden z připojených směrovačů zaslal zprávu Prune.

Generation ID je náhodná hodnota identifikující rozhraní. Podle ní může soused poznat, jestli se směrovač restartoval, v tom případně je nutné znovu ustavit spojení. State Refresh Capable slouží ke konfiguraci State Refresh Interval. RFC 3973 [3] popisuje přesnou skladbu položek pro výše uvedené typy podrobněji.

V případě, že směrovač dostane novou Hello zprávu od souseda, nastaví si časovač Livness Timer pro daného souseda na hodnotu z Hold Time pole. Pokud je přijata zpráva od nového souseda či se změnilo Generation ID, směrovač by měl odpovědět také Hello zprávu s náhodným zpožděním mezi 0 a 5 sekundami. Pole Hold Time se většinou nastavuje na 3,5 násobek Hello Timer, tedy 105 sekund. Je možné jej také nastavit na 0xffff, časovač pak nikdy nevyprší, nebo na 0 a časovač vyprší ihned. To se využívá při vypnutí rozhraní či změně IP adresy.

PIM Join/Prune zpráva

Join či Prune zprávu zasílá směrovač svému upstream sousedovi. Zpráva Prune se odesílá pro odříznutí větve od zdroje v momentě, kdy žádný z lokálních hostů směrovače už není součástí skupiny a o data ze zdroje již není zájem.

V případě, že downstream směrovač B již nemá zájem o zaslání dat, odešle svému upstream směrovači zprávu Prune. Pokud ale jiný downstream směrovač (A) má stále o multicast zájem, odpoví na Prune zprávu od B, zprávou Join. Je to jediný případ, kdy se zpráva Join používá v PIM-DM.

Zpráva (obrázek 2.3) obsahuje IP adresu upstream směrovače (Upstream Neighbor Address), počet multicastových skupin obsažených ve zprávě (Num Groups) a počet sekund (Hold Time), po které musí příjemce udržovat Prune stav. Následují skupiny polí pro každou multicastovou skupinu. Vždy obsahují adresu multicastové skupiny (Multicast Group Address), počet zdrojů obsažených ve zprávě, ke kterým se chce odesílatel připojit (Number of Joined Sources) nebo od kterých se chce odříznout (Number of Pruned Sources), a následují adresy těchto zdrojů (Join Source Address, Prune Source Address).

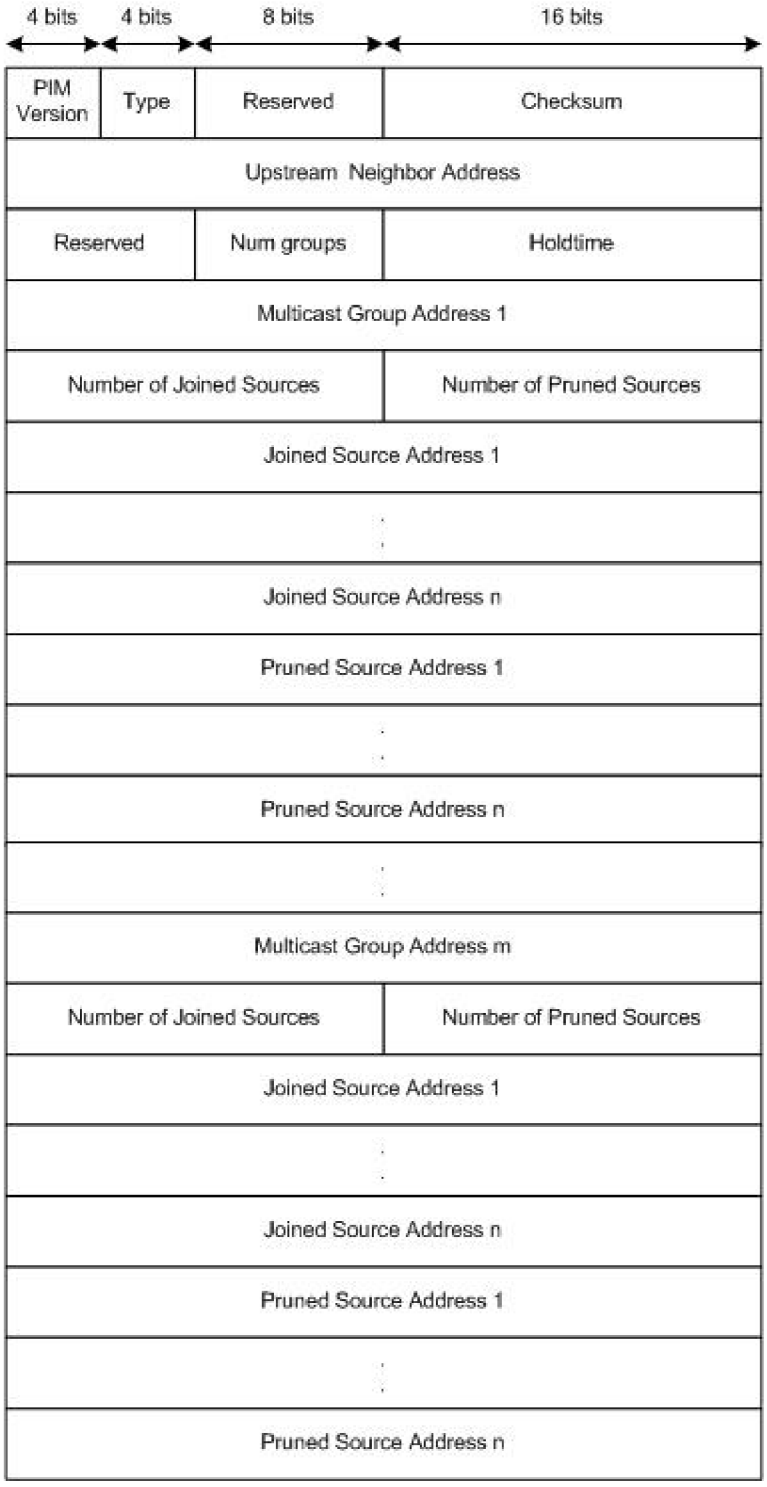
Všechny IP adresy využívané v Join/Prune, Assert, Graft a State Refresh jsou speciálně kódovány. Přesnější informace jsou uvedeny v [3].

PIM Assert zpráva

PIM Assert zpráva se používá, pakliže k jedné LAN síti je připojeno více směrovačů. Tento stav se projeví tak, že směrovač dostane multicastový datagram na downstream rozhraní (směrovače vidí multicast vyslaný od ostatních). Do LAN sítě ale musí vysílat multicast jen jeden z nich, proto všichni poté, co detekují tento stav, vyšlou Assert zprávu. Vyhrává směrovač s vyšší metrikou, případně vyšší IP adresou, pokud je metrika nerozhodná. [23]

Ve skutečnosti je to tak, že pokud přijde multicast na ne-RPF rozhraní, směrovač vyšle Prune zprávu. Pokud je ne-RPF rozhraní LAN, pošle se jako odpověď Assert zpráva. Ten kdo "bitvu"prohraje, odešle Prune zprávu na RPF rozhraní (pokud už multicast nepotřebuje). Assert zpráva se také zasílá jako odpověď na Assert zprávu od souseda.

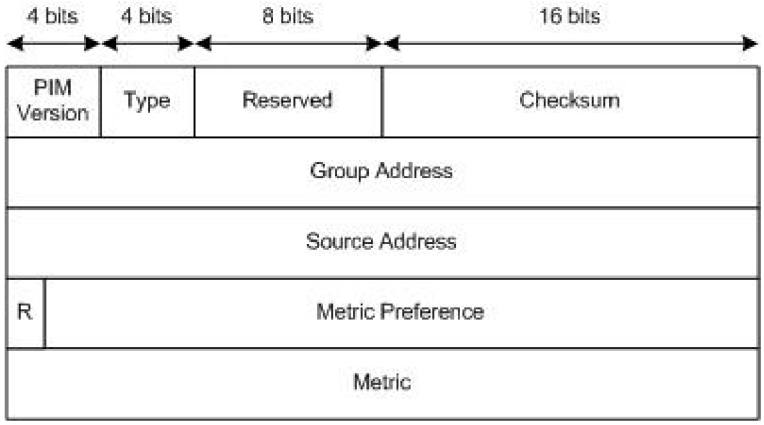
Zpráva Assert (obrázek 2.4) obsahuje adresu multicastové skupiny (Multicast Group Address) a zdroje (Source Address). Pole R (Rendezvous Point Tree bit) je pro PIM-DM nastaven na 0. Rozhodujícími hodnotami je metrika unicastové cesty ke zdroji (Metric) a administrativní vzdálenost unicastového směrovací protokolu pro případ, že směrovače používají jiný protokol (Metric Preference).



Obrázek 2.3: Formát PIM Join/Prune zprávy.

PIM Graft zpráva

Graft zprávu zasílá směrovač svému upstream sousedovi poté, co se v minulosti odřízl od stromu odesláním Prune zprávy, aby se ke stromu znovu připojil. Formát zprávy je stejný



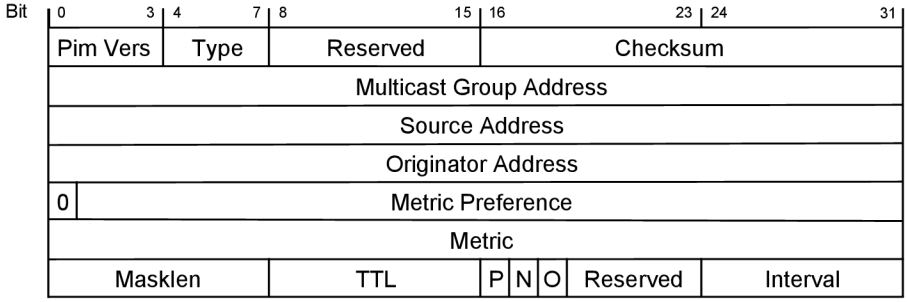
Obrázek 2.4: Formát PIM Assert zprávy.

jako u Join/Prune zprávy (obr. 2.3). Adresa zdroje musí být uvedena jako Join adresa a pole Hold Time je nastaveno na 0.

Jako odpověď na Graft zprávu se zasílá zpráva Graft Ack. Tato zpráva se odešle jen za předpokladu, že Graft zpráva byla doručena na některý z ne-RPF rozhraní. Formát Graft Ack zprávy je shodný s formátem Graft zprávy.

PIM State Refresh zpráva

PIM State Refresh zpráva je pravidelně rozesílána do sítě směrovačem s přímo připojeným zdrojem. Pokud ji příjme směrovač, který je ve stavu Prune, obnoví si svůj Prune časovač. Díky tomu nebude nutné síť neustále zaplavovat multicastem, jak je podstatou PIM-DM protokolu.



Obrázek 2.5: Formát PIM State Refresh zprávy.

Zpráva (obrázek 2.5) obsahuje adresu multicastové skupiny (Multicast Group Address), adresu zdroje (Source Address), adresu směrovače, který ji vyslal (Originator Address), administrativní vzdálenost (Metric Preference), metrika (Metric), délka masky unicastové cesty ke zdroji (Masklen) a TTL. Flagy O a N jsou ignorovány. Flag P je nastaven na 1, pokud je zasíláno na Pruned rozhraní, jinak 0. Interval je nastaven v sekundách mezi za sebou jdoucími zprávami State Refresh.

2.2.3 Stavy a stavové automaty

Multicastová směrovací tabulka je klíčová pro přeposílání přijatých multicastových dat. Její tvorba závisí na složité kombinaci konečných automatů, jejichž aktuální stav je ovlivněn mnoha faktory: přijaté zprávy, časovače, rozhraní, na kterém je zpráva přijata, atd. V protokolu PIM-DM se používají celkem čtyři stavové automaty.

Pro lepší názornost jsme všechny konečné automaty znázornili graficky. Kolečka jsou stavy a šipky mezi nimi přechody, počáteční stav je zvýrazněn. U každého přechodu je jedna či více očíslovaných podmínek, které mohou tento přechod vyvolat. Celá podmínka je podtržena černou čarou, pod čarou následují další akce, které je kromě přechodu, nutné vykonat. V diagramech jsou použity speciální značky pro zkrácení zápisu, které jsou vysvětleny blíže v legendě (obrázek 2.6).

Legenda
→✉ {Nazev}..... Přijmutí zprávy typu {Nazev} na rozhraní I/RPF
✉→ {Nazev}..... Odeslání zprávy typu {Nazev} na rohraní I/RPF
🕓 {Nazev}..... Časovač typu {Nazev}
*... S is not directly connected
Preferred {Nazev}... Zprava {Nazev} s lepší metrikou než current winner
Inferior {Nazev}..... Zprava {Nazev} s horší metrikou než current winner

Obrázek 2.6: Legenda pro konečné automaty.

U každého konečného automatu je také přepis pomocí tabulky. Každý řádek prezentuje nějakou událost, každý sloupec prezentuje stav, ve kterém se automat nacházel před událostí. Symbol šipky znamená přechod do stavu, který je označen počátečním písmenem (→P znamená "Přechod do stavu Pruned").

2.2.4 Upstream (S,G) State Machine

Tento automat (obrázek 2.7, tabulka 2.1) existuje pro každou dvojici (S,G) a ovlivňuje stav upstream rozhraní. Může se nacházet ve třech stavech: Forwarding, Pruned a AckPending. Automat ovlivňují zprávy Pruned, Join a Graft.

Forwarding je počáteční stav automatu, v tomto stavu se automat bude také nacházet, dokud nebude oilist (viz konec subsekce 2.2.1) prázdný. V případě, že oilist je prázdný, automat bude ve stavu Pruned. AckPending je přechodný stav mezi Pruned a Forwarding, kdy se čeká na přijetí zprávy Graft Ack.

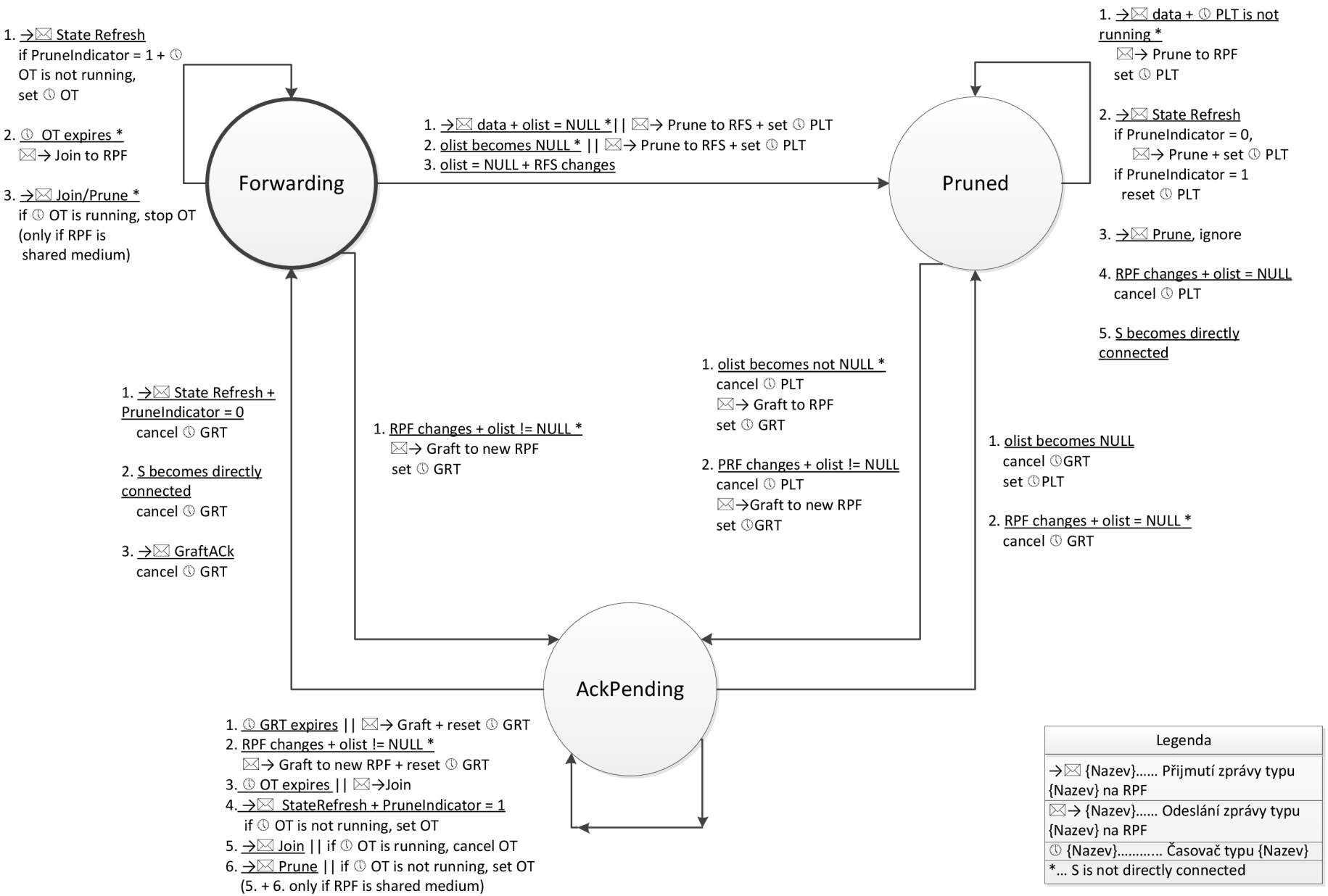
Automat pracuje se třemi časovači. Časovač Override Timer (OT(S,G)) se nastaví v případě příjmu zprávy Pruned (některý ze sousedů na segmentu již nemá zájem o multicast). Časovač se používá z toho důvodu, aby nedošlo k zaplavení upstream souseda zprávami Join. Zprávu Join, která zabrání odříznutí směrovače, může zaslat až po jeho vypršení. Jeho hodnota je náhodná v rozmezí 0 a Override_interval (OI(I)). OI je implicitně 2,5 sekundy.

Časovač Prune Limit Timer PLT(S,G) se používá pouze ve stavu Pruned. Dokud časovač běží, směrovač nemůže na upstream rozhraní zasílat zprávy Pruned. Tento časovač také zabraňuje vzniku záplavy zpráv. Časovač se nastavuje na 210 sekund.

Graft Retry Timer (GRT(S,G)) se nastaví při přechodu do AckPending na 3 sekundy. Pokud v tomto intervalu nepřijde na upstream rozhraní zpráva Graft Ack, směrovač musí znovu vyslat zprávu Graft.

Událost	**Forwarding**	**Pruned**	**AckPending**
Přijme data RPF AND olist(S,G) == NULL AND časovač PLT(S,G) neběží	→P, odešle Prune(S,G) a nastaví PLT(S,G)	→P, odešle Prune(S,G) a nastaví PLT(S,G)	N/A
Přijme State Refresh(S,G) na RPF AND Prune Indicator == 1	→F, nastaví OT(S,G)	→P, obnoví PLT(S,G)	→AP, nastaví OT(S,G)
Přijme State Refresh(S,G) na RPF AND Prune Indicator == 0 AND PLT(S,G) neběží	→F	→P, odešle Prune(S,G), nastaví PLT(S,G)	→F, zruší GRT(S,G)
Přijme Join(S,G) na RPF	→F, zruší OT(S,G)	→P	→AP, zruší OT(S,G)
Přijme Prune(S,G)	→F, nastaví OT(S,G)	→P	→AP, nastaví OT(S,G)
OT(S,G) vyprší	→F, odešle Join(S,G)	N/A	→AP, odešle Join(S,G)
olist(S,G)→NULL	→P, odešle Prune(S,G), nastaví PLT(S,G)	N/A	→P, odešle Prune(S,G), nastaví PLT(S,G), zruší GRT(S,G)
olist(S,G)→non-NULL	N/A	→AP, odešle Graft(S,G), nastaví GRT(S,G)	N/A
RPF se změní AND olist(S,G) != NULL	→AP, odešle Graft(S,G), nastaví GRT(S,G)	→AP, odešle Graft(S,G), nastaví GRT(S,G)	→AP, odešle Graft(S,G), nastaví GRT(S,G)
RPF se změní AND olist(S,G) == NULL	→P	→P, zruší PLT(S,G)	→P, zruší GRT(S,G)
Zdroj S se přímo připojí	→F	→P, zruší PLT(S,G)	→F, zruší GRT(S,G)
GRT(S,G) vyprší	N/A	N/A	→AP, odešle Graft(S,G), nastaví GRT(S,G)
Příjme Graft Ack(S,G) na RPF	→F	→P	→F, zruší GRT(S,G)

Tabulka 2.1: Upstream (S,G) State Machine v tabulkové podobě



Obrázek 2.7: Upstream (S,G) State Machine.

2.2.5 Downstream (S,G,I) State Machine

Tento automat existuje pro každou trojici (S,G,I[2]). Může se nacházet ve třech stavech: NoInfo, Pruned a PrunePending. Na upstream rozhraní je vždy ve stavu NoInfo. Automat ovlivňují zprávy Pruned, Join a Graft.

Počátečním stavem automatu (obrázek 2.8, tabulka 2.2) je NoInfo. V tomto stavu neběží žádný časovač. Z tohoto stavu je možné přejít pouze do přechodného stavu PrunePending přijetím zprávy Prune. Ve stavu PrunePending směrovač čeká, zda nebude zpráva Prune anulována nějakou zprávou Join. Pokud anulován není, přejde automat do stavu Pruned. V tomto stavu směrovač nezasílá na rozhraní multicast.

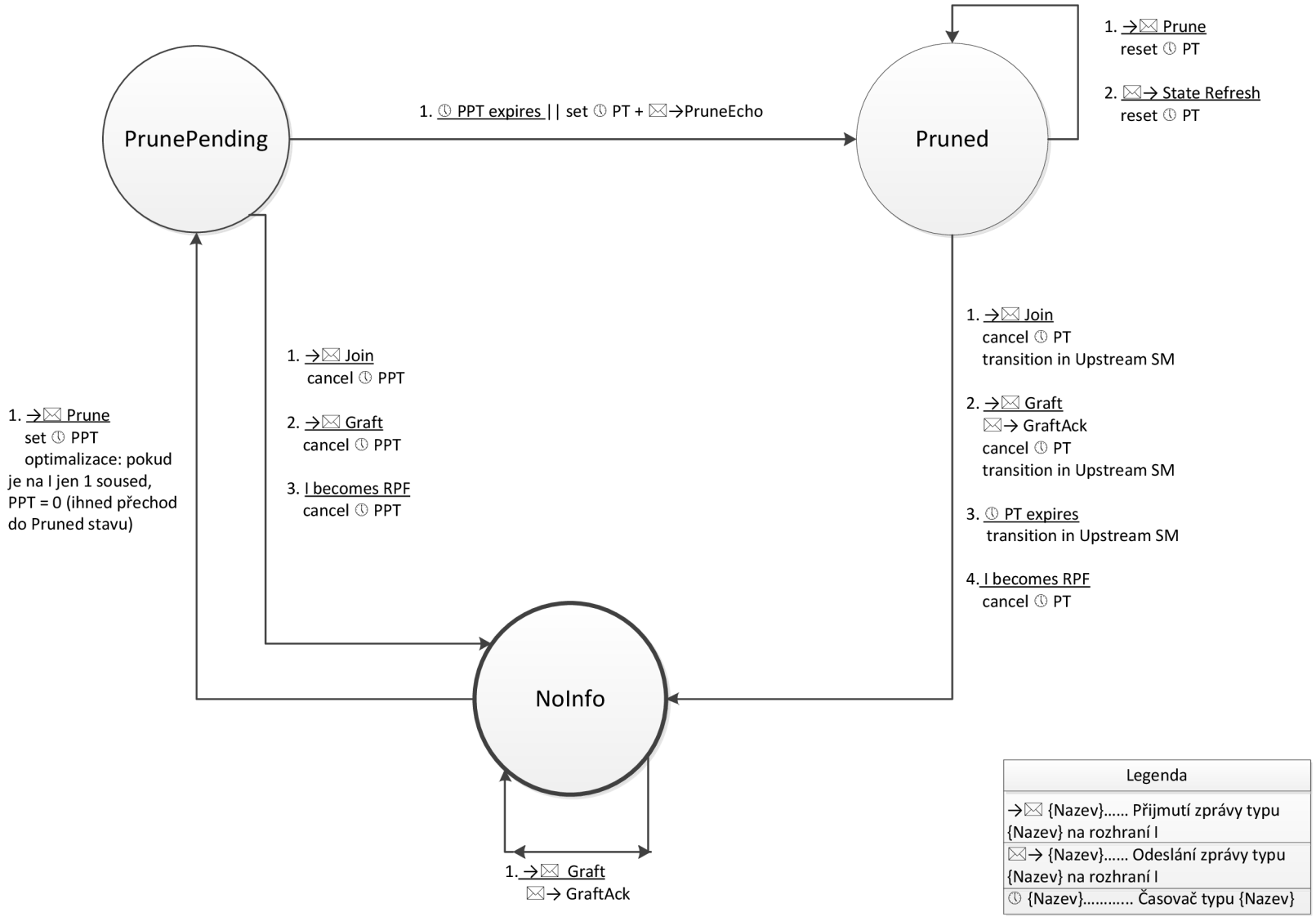
Stav automatu ovlivňují také dva časovače. Časovač Prune Pending Timer se nastaví přechodem do stavu PrunePending. Pokud do doby OI(I) + PD(I) nepřijde zpráva Join, automat přejde do stavu Pruned. PD(I) má implicitně hodnotu 0,5 sekund.

Časovač Prune Timer se nastaví při přechodu do stavu Pruned na základě Hold Time hodnoty ve zprávě Prune. Po jeho vypršení přejde automat zpět do počátečního stavu NoInfo.

Událost	**No Info**	**Prune Pending**	**Pruned**
Přijme Prune(S,G)	→PP, nastaví PPT(S,G,I)	→PP	→P, obnoví PT (S,G,I)
Přijme Join(S,G)	→NI	→NI, zruší PPT(S,G,I)	→NI, zruší PT(S,G,I)
Přijme Graft(S,G)	→NI, odešle Graft Ack	→NI, odešle Graft Ack, zruší PPT(S,G,I)	→NI, odešle Graft Ack, zruší PT(S,G,I)
Časovač PPT(S,G) vyprší	N/A	→P, nastaví PT(S,G,I)	N/A
Časovač PT(S,G) vyprší	N/A	N/A	→NI
RPF se změní na I	→NI	→NI, zruší PPT(S,G,I)	→NI, zruší PT(S,G,I)
Odešle State Refresh(S,G) z I	→NI	→PP	→P, obnoví PT(S,G,I)

Tabulka 2.2: Downstream (S,G,I) State Machine v tabulkové podobě

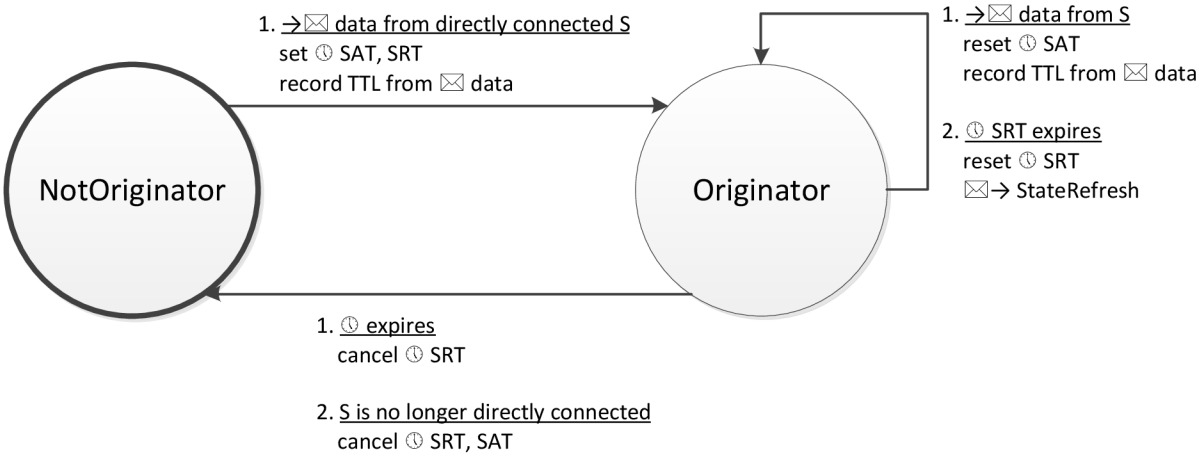
[2] I = rozhraní.



Obrázek 2.8: Downstream (S,G,I) State Machine.

2.2.6 Origination State Machine

Zda bude směrovač tvůrcem zpráv State Refresh (viz 2.2.2) závisí na Origination konečném automatu (obrázek 2.9, tabulka 2.3), který existuje na směrovači pro každý zdrojový strom (S,G). Má pouze dva stavy NotOriginator a Originator.



Obrázek 2.9: Origination State Machine.

Událost	**NotOriginator**	**Originator**
Přijme data od zdroje a zdroj je přímo připojený	→O, nastaví SRT(S,G), nastaví SAT(S,G)	→O, obnoví SAT(S,G))
SRT(S,G) vyprší	N/A	→O, odešle StateRefresh(S,G), obnoví SRT(S,G)
SAT(S,G) vyprší	N/A	→NO, zruší SRT(S,G)
Zdroj již není přímo připojený	→NO	→NO, zruší SRT(S,G), zruší SAT(S,G)

Tabulka 2.3: State Refresh State Machine v tabulkové podobě

NotOriginator je počáteční stav, ve kterém směrovač nemá povinnost vysílat State Refresh zprávy (samozřejmě zprávy, které přijme od svého upstream souseda, přeposílá dále). V případě, že zjistí, že zdroj S je přímo připojen ke směrovač, přejde automat do stavu Originator, ve kterém musí v pravidelných intervalech vysílat State Refresh zprávy.

Vždy po uplynutí State Refresh Timer (SRT(S,G)) směrovač ve stavu Originator vytvoří a vyšle novou zprávu State Refresh. Pokaždé, kdy přijdou od zdroje multicastové datagramy, resetuje se časovač Source Active Timer (SAT(S,G)) na hodnotu 210 sekund. Pokud SAT vyprší, směrovač se vrátí do stavu NotOriginator.

2.2.7 Assert Message (S,G) State Machine

Automat (obrázek 2.10, tabulka 2.4) pracující se zprávami Assert Message (viz 2.2.2) existuje pro každé rozhraní a zdrojový strom. Může se nacházet ve stavu NoInfo, Winner nebo Loser.

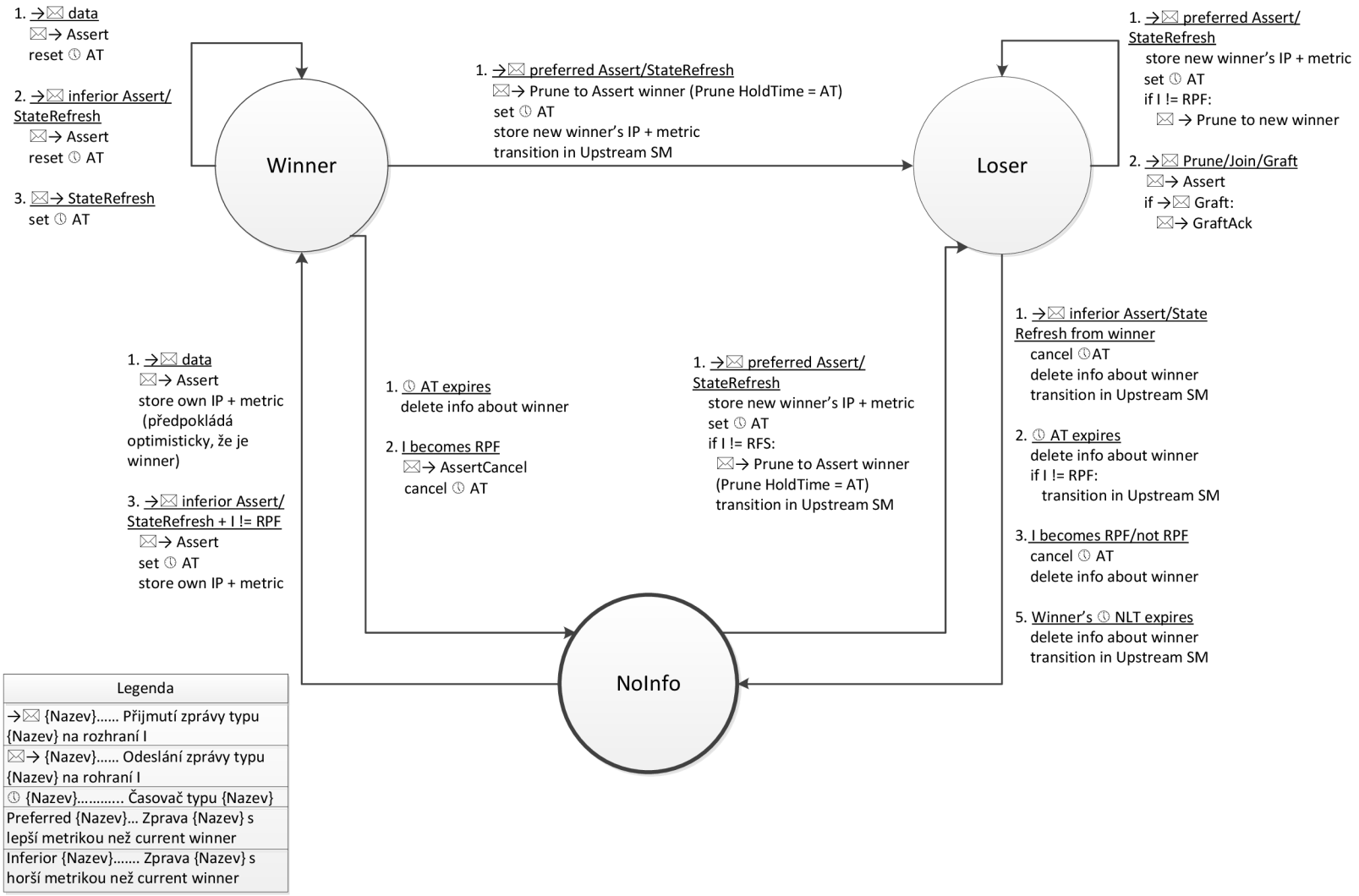
Událost	**No Info**	**Winner**	**Loser**
Přijme (S,G) data na downstream rozhraní	→W, odešle Assert(S,G), nastaví AT(S,G,I)	→W, odešle Assert(S,G), nastaví AT(S,G,I)	→L
Přijme horší[3] (Assert OR State Refresh) od Assert vítěze	N/A	N/A	→NI, zruší AT(S,G,I)
Přijme horší (Assert OR State Refresh) od Assert poraženého AND CouldAssert == TRUE	→W, odešle Assert(S,G), nastaví AT(S,G,I)	→W, odešle Assert(S,G), nastaví AT(S,G,I)	→L
Příjme preferovaný[4] Assert OR State Refresh	→L, odešle Prune(S,G), nastaví AT(S,G,I)	→L, odešle Prune(S,G), nastaví AT(S,G,I)	→L, nastaví AT(S,G,I)
Odešle State Refresh	→NI	→W, obnoví AT(S,G,I)	N/A
AT(S,G) vyprší	N/A	→NI	→NI
CouldAssert → FALSE	→NI	→NI, zruší AT(S,G,I)	→NI, zruší AT(S,G,I)
CouldAssert → TRUE	→NI	N/A	→NI, zruší AT(S,G,I)
Vítězův NLT(N,I) vyprší	N/A	N/A	→NI, zruší AT(S,G,I)
Příjme Prune(S,G) OR Join(S,G) OR Graft(S,G)	→NI	→W	→L, odešle Assert(S,G)

Tabulka 2.4: Assert Message (S,G) State Machine v tabulkové podobě

Počátečním stavem je NoInfo, rozhraní nemá žádný Assert stav. Pakliže směrovač zvítězí v Assert vyjednávání, nachází se ve stavu Winner. V tomto stavu je zodpovědný za zasílání multicastu (S,G) na rozhraní I. V případě, že ve vyjednávání prohraje, automat bude v stavu Loser. V tomto stavu směrovač nesmí zasílat multicast (S,G) na rozhraní I.

V automatu se objevuje zpráva Assert Cancel. Tu zasílá vítěz, jehož rozhraní I se stalo rozhraním RPF, pouze kvůli optimalizaci. Bylo by také alternativně možno počkat, až vyprší časovač AT. Ve zprávě Assert Cancel se posílá metrika s největšími možnými hodnotami (simuluje nekonečno), čímž se ihned iniciuje Assert vyjednávání.

[3]Horší assert je takový, který má horší metriku než daný směrovač.
[4]Preferovaný assert je takový, který má lepší metriku než aktuální vítěz.



Obrázek 2.10: Assert Message (S,G,I) State Machine.

2.3 Protocol Independent Multicast - Sparse Mode (PIM-SM)

PIM-SM [11] je multicastový směrovací protokol, který využívá unicastové směrovací informace jako základ pro šíření multicastových datagramů na všechny multicastové směrovače, které mají o data zájem. To je zásadní rozdíl oproti PIM-DM, který posílá data všem bez ohledu na to, zda o data mají zájem či ne.

2.3.1 Základní informace

PIM-DM má hodně společného s PIM-SM. Oba protokoly využívají pro multicastové směrování tabulky MRIB, TIB a MFIB. Také zprávy užívané v PIM-SM jsou spíše rozšířením zpráv PIM-DM. Assert volba je shodná s tou z předešlého protokolu. Přestože PIM-SM může také vytvářet a vytváří zdrojové stromy, jsou pro něj ale typické spíše stromy sdílené.

Se sdíleným stromem je také spjatý pojem Rendezvous Point (RP). Je to směrovač, který je nakonfigurován, aby se stal kořenem sdíleného stromu pro určitou multicastovou skupinu. Proto se také zdrojové stromy označují (*,G), nevedou totiž ke zdroji dat, ale k směrovači, který všechny zdroje sjednocuje. Příjemci zasílají Join/Prune zprávy směrovači RP a data jsou k nim zasílaná také z RP.

Aby směrování mohlo správně fungovat, příjemci musí znát IP adresu RP. Ta může být manuálně nakonfigurována anebo je možné ji zjistit dynamicky pomocí Bootstrap Router (BSR) mechanismu (viz 2.3.4).

Novým pojmem je také Designated Router (DR). Na sdíleném LAN médiu jako je ethernet, může být připojeno více PIM-SM směrovačů. Jen jeden z nich, DR, se ale bude pro lokální hosty tvářit jako PIM-SM směrovač a bude zajišťovat všechny služby spojené s PIM-SM. Pro každé rozhraní se zvolí jednoduchou volbou vždy jeden DR směrovač.

PIM-SM směrování má tři fáze, ve kterých se vytvoří potřebné stromy, po kterých se budou vysílat data od zdroje S k příjemcům G. Pro korektní fungování PIM-SM postačí jen 1. fáze. Kvůli optimalizaci, ale následuje ještě 2. a 3 fáze.

1. fáze: RP strom

Nejprve se pro každou LAN síť volí Designated Router (DR). Na síti LAN může být připojeno více směrovačů, ale jen jeden může zajišťovat multicastová data pro určitou skupinu. DR se dozví od lokálních hostů, kdo má zájem o multicast pomocí protokolu IGMP či MLD (pro IPv6).

Pokud se některý z hostů chce připojit do skupiny, DR pošle zprávu (*,G) Join směrem k RP směrovač, případně k prvnímu směrovač po cestě, který je v (*,G) Join stavu. Na základě zpráv Join, které zasílají směrovače DR k RP se vytvoří sdílený strom (*,G), také nazývaný RP strom.

Zprávy Join zasílají DR v pravidelných intervalech, pokud mají zájemce o multicast. Jakmile DR zjistí, že žádný z hostů již o data nemá zájem, vyšle směrem k RP zprávu Prune, čímž je část sítě odříznuta od sdíleného stromu.

Na druhé straně zdroj začne vysílat multicast. Lokální DR je zabalí do PIM Register zprávy a zašle je unicastově směrovači RP. Ten je rozbalí a posílá je dál podél sdíleného stromu všem příjemcům ze skupiny G.

2. fáze: Register-Stop

Proces registrace je značně neefektivní. Je nutné data zabalovat a rozbalovat, což nás stojí výpočetní prostředky. Pokud je zdroj někde blízko příjemců, je také velmi neefektivní, aby data putovali k RP a pak zpět k příjemci.

Z tohoto důvodu RP vyšle zprávu (S,G) Join směrem ke zdroji, aby vytvořil zdrojový strom (S,G). Data tak mohou nativně putovat od zdroje S k RP. Pokud data narazí na směrovač, který je také v (*,G) Join stavu, dojde ke zkrácení cesty přímo k příjemcům.

V tento okamžik ale RP dostává data nativně přes zdrojový strom i zabalená v PIM Register zprávě. RP proto vyšle k DR zdroje S zprávu Register-Stop, čímž zastaví proces registrace.

3. fáze: Shortes-Path Tree

Ani po ukončení 2. fáze není trasa vedoucí přes RP pro všechny příjemce optimální. Optimální není, pokud unicastová cesta mezi příjemcem a zdrojem S je kratší, než cesta přes RP. Z tohoto důvodu může DR vyslat také (S,G) Join zprávu ke zdroji a stát se tak součástí zdrojového stromu nejkratší cesty SPT (Source-Specific Shortest-Path Tree).

(S,G) Join dorazí až ke zdroji nebo k jinému směrovač, který je v (S,G) Join stavu. K příjemci začnou putovat multicastová data nejkratší možnou cestou. Avšak stále příjemce dostává data i přes RP. DR proto odešle (S,G) Prune zprávu směrem k RP, nazývá se také (S,G,rpt) Prune. Tím se od RP odřízne.

Volba DR směrovače

Tato volba je poměrně jednoduchá a je závislá na zprávách Hello. Na rozdíl od PIM-DM, PIM-SM využívá ve zprávě i pole DRPriority. Zvýšením implicitní priority může síťový administrátor nakonfigurovat, který směrovač bude zvolen DR. Implicitní hodnota je 1.

Každý směrovač si ke svému sousedovi ukládá informace z Hello zprávy. Jedná se o rozhraní, na které zpráva přišla, primární IP adresa souseda, DR priorita, booleovskou hodnotu přítomnosti DR priority a časovač NLT (Neighbor Liveness Timer) z pole Hold Time.

Při volbě DR směrovače projde směrovač všechny sousedy na rozhraní a porovná jejich priority. Vyhrává ten s větší prioritou. Pokud ale priorita není ve zprávě přítomna nebo jsou priority shodné, vyhrává ten s vyšší IP adresou.

Volba DR směrovače není trvalá, mění se na základě přicházejících Hello zpráv. DR priorita se totiž může v čase měnit. Stejně tak se může připojit nový směrovač nebo jiný vypadnout. Ostatní směrovače na tuto situaci dokážou reagovat a opakovat volbu.

2.3.2 PIM-SM zprávy

PIM-SM používá tyto zprávy:

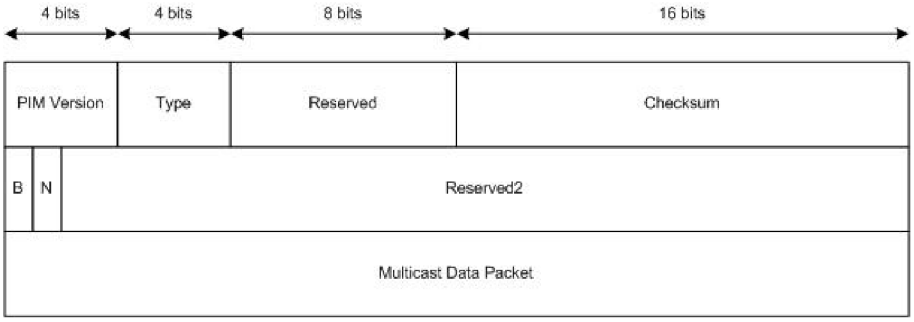
- Hello Message
- Register Message
- Register-Stop Message
- Prune Message
- Join Message

- Assert Message

Mnohé z nich jsme si už vysvětlili u protokolu PIM-DM (viz 2.2.2), proto se zde zaměříme jen na ty, které nově přibyly.

PIM Register zpráva

Jak již bylo uvedeno v 2.3.1 v 1. fázi ustavení multicastového provozu, zasílá DR zdroje data směrovači RP zabalené ve zprávě Register (obrázek 2.11). Stejně tak tuto zprávu může používat PMBR[3] při zasílání multicastu do RP stromu.



Obrázek 2.11: Formát PIM Register zprávy.

Zdrojová IP adresa je nastavena na IP adresu DR. Border bit B nastaví DR směrovač na 0. Pokud je odesílatelem PMBR nastaví jej na 1. Null-Register bit N je ve většině případů nastaven na 0. Pokud je nastaven na 1, označuje tzv. Null-Register zprávu. Tu DR odešle směrovači RP těsně před vypršením časovače Register-Stop Timer, aby RP měl možnost obnovit Register-Stop stav na směrovači DR zasláním nové zprávy Register-Stop. Pokud tak RP neučiní a časovač Register-Stop Timer vyprší, DR začne znovu zasílat data zabalená do Register zprávy.

V poli Multicast data packet se nachází data vysílaná zdrojem. Před zabalením se musí dekrementovat TTL, to samé musí provést RP po rozbalení dat. V případě Null-Register zprávy je toto pole prázdné.

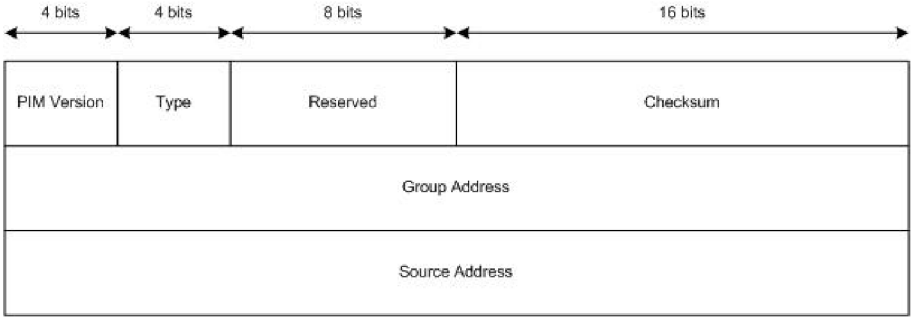
PIM Register-Stop zpráva

Register-Stop zpráva (obrázek 2.12) se využívá v 2. fázi (2.3.1) ustavení multicastového provozu. Zprávu unicastově vysílá RP směrovači DR (zdroj Register zpráv). DR po obdržení přestane posílat data zabalená do Register zprávy, ale začne je zasílat nativně. Zpráva obsahuje, kromě hlavičky, pouze adresu skupiny (Group Address) a adresu zdroje (Source Address).

2.3.3 Stavy a stavové automaty

Problematika stavů a stavových automatů je v protokolu PIM-SM mnohonásobně složitější než v případě PIM-DM. PIM-SM pracuje s celkem 11 konečnými automaty. Pro větší přehlednost se celá TIB tabulka dělí na 4 sekce:

[3]PMBR = PIM Multicast Border Router, umožňuje propojení více PIM domén



Obrázek 2.12: Formát PIM Register-Stop zprávy.

- (*,*,RP) stav (pro všechny stromy obsluhované jedním RP),
- (*,G) stav (pro každý RP strom),
- (S,G) stav (pro každý zdrojový strom),
- (S,G,rpt) stav (pro každou skupinu, pro kterou existuje zdrojový strom (S,G) i sdílený strom (*,G)).

Pro každý stav existují upstream a downstream konečné automaty, pro stavy (S,G) a (*,G) existují ještě Assert konečné automaty a DR směrovače mají navíc Register konečný automat.

2.3.4 Bootstrap Router mechanismus

Všechny směrovače, které se účastní směrování multicastového provozu, musí znát IP adresu RP. Statická konfigurace není při větším množství směrovačů příliš efektivní. Z toho důvodu definuje RFC 5059[4] Bootstrap Router (BSR) mechanismus, který umožňuje konfiguraci dynamickou.

V síti jsou některé směrovače nakonfigurovány jako Candidate-RP a některé jako Candidate-BSR. Jeden ze směrovačů pak bude zvolen BSR. Ten vybere ze všech kandidátů RP směrovač a následně bude distribuovat informaci o mapování RP na multicastové skupiny všem směrovačům v PIM doméně. Celý proces má 4 kroky:

1. Volba BSR: BSR kandidáti začnou zaplavovat doménu Bootstrap zprávami. Každá obsahuje prioritu. Pokud má některý z BSR kandidátů nižší prioritu než tu, která mu přišla od jiného kandidáta ve zprávě, přestane se účastnit volby. Nakonec zbyde pouze jeden kandidát, který se stane BSR směrovačem a tuto informaci rozhlásí po doméně Bootstrap zprávou.
2. Ohlášení Candidate-RP: RP kandidáti periodicky zasílají zprávu Candidate RP Advertisement zvolenému BSR. Zpráva obsahuje prioritu RP a seznam multicastových skupin, pro které směrovač kandiduje.
3. Vytvoření RP množiny: BSR vybere podmnožinu z RP kandidátů, kteří mu zaslali zprávu o své kandidatuře. Výsledná množina nesmí být příliš velká, aby nebylo náročné ji distribuovat všem směrovačům v doméně, ani příliš malá, aby RP nebyly příliš vytížené.

4. Šíření RP množiny: BSR periodicky vysílá Bootstrap zprávy, které se šíří do celé domény a nově obsahují RP množinu.

2.4 Protocol Independent Multicast - Source Specific Mode (PIM-SSM)

PIM-SSM je multicastový směrovací protokol založený na modelu SSM (Source Specific Multicast). Model SSM je popsán v RFC 3569[5].

2.4.1 Základní koncept

U tradičního multicastového modelu Any-Source Multicast (ASM) (RFC 1112), někdy také označováno jako Internet Standard Multicast (ISM) jsou multicastové skupiny identifikovány pouze multicastovou adresou skupiny. Zdrojem do takové skupiny pak můžou být jakékoliv počítače, které mohou či nemusí být součástí skupiny. Koncový uživatel tedy nemůže ovlivnit, od koho mu data budou přicházet.

Naopak u PIM-SSM je multicastová skupina identifikována nejen adresou skupiny, ale i zdrojem. SSM multicastová skupina se označuje jako kanál (S,G) (oproti tradičnímu (*,G)), kde S je adresa zdroje a G adresa skupiny. Pro SSM skupiny jsou vyhrazeny adresy 232.0.0.0/8 (IPv4) a FF3x::/32 (IPv6).

Přihlášení ke kanálu je podporováno pouze protokolem IGMPv3 [7] (pro IPv4), případně MLDv2 [22] (pro IPv6). Protokol IGMPv2 a IGMPv1 totiž neumožňují specifikovat adresu zdroje multicastu. Na rozdíl od PIM-SM se nevytváří žádné sdílené stromy, ale pouze stromy zdrojové, přímo od zdroje S k hostu H ze skupiny G. SSM také nepotřebuje RP.

2.4.2 Výhody oproti ASM

Architektura ASM má několik problému a nevýhod, které se mnohem flexibilnější architektura SSM snaží odstranit. [5]

Alokace multicastových adres

ASM nevládne technikou, která by umožnila zabránit adresním kolizím. U IPv6 není tento problém tak markantní, přece jen aplikace mají větší výběr v adresách. U IPv4 však není rozsah multicastových adres tak velký. Částečným, ale nedostačujícím řešením může být GLOP[15] či Multicast Address Allocation Architecture ([20]).

Vzhledem k tomu, že SSM kanály (S1, G) a (S2, G) jsou odlišné a nepřekrývají se, nemůže ani dojít ke kolizi multicastových adres.

Řízení přístupu

Základní problém, který vychází z podstaty ASM, je ten, že neumožňuje specifikovat zdroj, od kterého bude příjemce přijímat data. I když už je vybudován strom ke zdroji, host nemá žádnou záruku, že data nezačne do stejné skupiny vysílat i někdo jiný.

U SSM se uživatel přihlašuje ke kanálu (S,G), a tak má zaručeno, že data nebude dostávat od nikoho jiného než od zdroje s adresou S (pomineme-li možnosti IP spoofingu a jiných útoků).

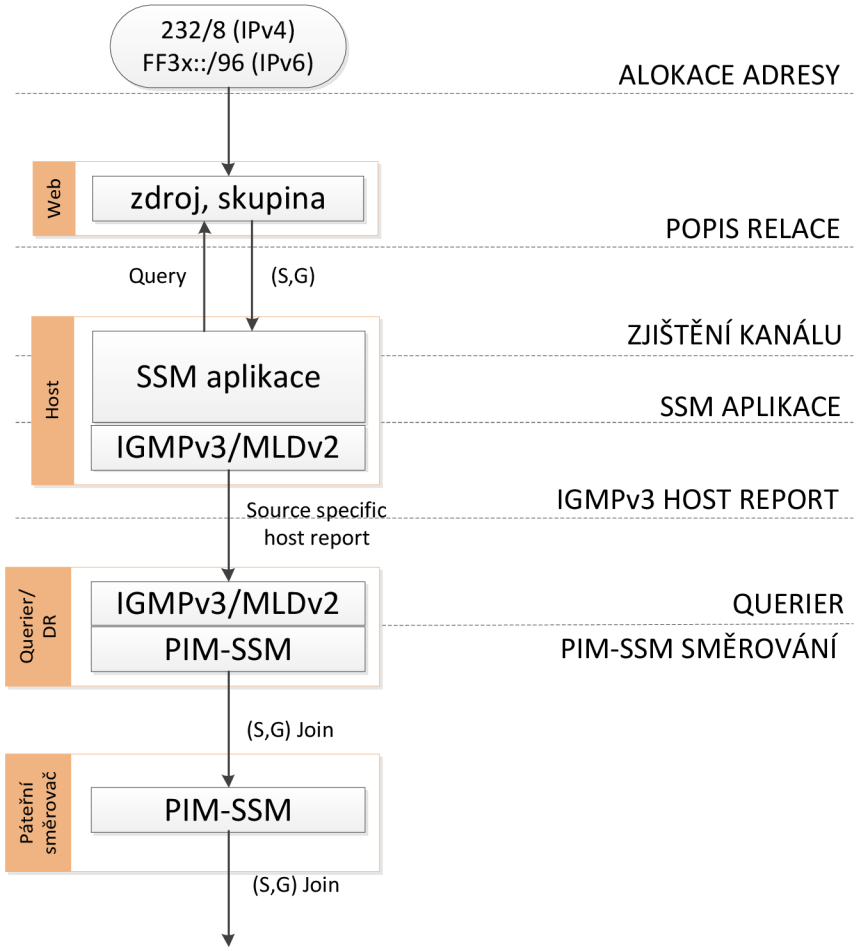
Obsluha dobře známých zdrojů

PIM-SM se nechová příliš efektivně, pokud je adresa zdroje dobře známá (well-known address). Směrování pomocí sdíleného stromu nemusí být pro příjemce efektivní, někdy může být přímá cesta (shortest forwarding path) od zdroje k hostovi mnohem efektivnější a sdílený strom postrádá smysl.

SSM využívá pouze zdrojové stromy. Není nutné tedy volit RP, vytvářet sdílené cesty, ani využívat protokol MSDP [4]. Z toho vyplývá, že SSM má méně komplexní infrastrukturu než ASM.

2.4.3 Architektura SSM

SSM framework můžete vidět na obrázku 2.13. Ten ilustruje základní části, ze kterých se architektura SSM skládá. K jednotlivým částem jsou dostupné další RFC, které je dopodrobna popisují.



Obrázek 2.13: SSM framework.

[4] MSDP (Multicast Source Discovery Protocol) je protokol z rodiny PIM směrovacích protokolů, který umožňuje propojit více PIM-SM domén, což mimo jiné zajišťuje RP redundanci.

V 2.4.2 jsme zmínili několik výhod SSM módu. Je tedy na místě zmínit i jednu nevýhodu. Všichni příjemci multicastu musí znát IP adresu zdroje, jinak se nemohou do multicastové skupiny zapojit. Toto je možné řešit manuální konfigurací nebo publikováním na webových stránkách.

Dále je nutné, aby host měl nainstalovanou SSM aplikaci. Taková aplikace musí umět zjistit adresu kanálu skládající se ze zdrojové i cílové IP adresy. Bližší informace k API najdete v [19]. Host musí také podporovat pro připojení ke skupině protokol IGMPv3 (případně MLD2).

V dnešní době se stále nejčastěji používá PIM-SM. Proto je nutné zajistit určitou koexistenci s SSM. Předně pokud DR dostane žádost (S,G), kde G je z SSM rozsahu, musí iniciovat Join(S,G) a ne Join(*,G). Páteřní směrovače nesmí propagovat Join(*,G), pokud je G z SSM rozsahu. RP nesmí přijmout zprávu PIM Register nebo Join(*,G) zprávu, pokud se G nachází v SSM rozsahu. Přesto určitá malá podmnožina protokolu PIM-SM je pro provoz SSM nutná.

SSM nevnáší do IP multicastu žádné nové bezpečnostní opatření. Dokáže ale pomoci v prevenci proti DoS[5] útoku, který může nastat celkem jednoduše, pokud nemůžeme ovlivnit zdroj dat (v ASM). Avšak existují možnosti, jak toto obejít.

2.5 Bidirectional Protocol Independent Multicast (BiDiR-PIM)

PIM-SM vytváří jednosměrný sdílený strom, který se používá pouze pro přenos dat od RP k příjemcům. Protokol BiDiR-PIM, specifikovaný v RFC 5015[13], je rozšířením PIM-SM, které vytváří obousměrný sdílený strom mezi zdroji a příjemci multicastové skupiny. Tím umožňuje zasílání dat v obou směrech a nepotřebuje budovat zdrojový strom, což přináší mnoho výhod.

2.5.1 Obousměrný sdílený strom

Připomeňme si, jakým způsobem zdroje zasílají svá data směrovači RP v případě využití PIM-SM. Existují v podstatě dvě možnosti. Nejprve směrovač přímo přepojený ke zdroji zabalí data do registrační zprávy a unicastem je vyšle přímo k RP směrovači. Data se musí na směrovači rozbalit a následně distribuovat sdíleným stromem. V druhé fázi se vybuduje zdrojový strom od zdroje k RP. Tento strom se využívá k nativnímu zasílání dat do skupiny.

Oba jmenované způsoby přináší problémy. Při balení dat do registrační zprávy dochází ke zbytečné práci jak na straně DR směrovače u zdroje, tak u RP směrovače, který musí zprávu znovu rozbalit. Zvyšuje se také zpoždění a velikost zasílaných dat. Vybudování zdrojového stromu je také náročné na (zejména paměťové) zdroje směrovačů.

Protokol BiDiR-PIM vznikl, aby tyto problémy řešil a umožnil nativní zasílání dat bez nutnosti výstavby zdrojových stromů. Data od zdroje jsou zasílána vzhůru sdíleným stromem k RP, který je kořenem stromu. Odsud putují stromem dolů k příjemcům v každé větvi stromu.

Při zasílání dat od RP k příjemcům je postup zcela shodný s protokolem PIM-SM. Tento protokol ale neumožňuje, aby byla sdíleným stromem zasílána data také k RP. Data jsou totiž akceptována pouze při příchodu na RPF rozhraní, což zabraňuje vzniku smyček. Aby

[5]Denial of Service (Dos) je typ útoku, jehož cílem je znepřístupnit počítače nebo síťove zdroje jejich běžným uživatelům.

smyčky nevznikaly v BiDir-PIM při zasílání dat směrem k RP, představuje protokol nový mechanismus - volba Designated Forwarder (DF) směrovače.

2.5.2 Volba DF směrovače

DF směrovač se volí na každé lince, ať už je typu multiaccess (více přístupová) nebo point-to-point (dva síťové uzly). Volba se provádí v době, kdy se automaticky zjišťují i adresy RP směrovačů. Pro každý RP nebo přesně RP adresu se na lince volí jeden DF směrovač. BiDir-PIM umožňuje použít RP adresu (RPA), která není fyzicky přiřazena žádnému směrovači. Pracuje proto s pojmem Rendezvous Point Link (RPL), což je fyzická linka, ke které RPA přísluší. Na RPL k DF volbě nedochází.

DF směrovačem se stává ten, který má nižší unicastovou metriku pro cestu k RP směrovači. Informace o metrice se zasílá v speciálních zprávách (DF Election Packet) pro DF volbu. Směrovače na RPF rozhraní vysílají zprávy s metrikou rovnu nekonečnu. V případě, že dojde ke změně metriky, DF umře nebo přibyde nový směrovač na lince, dochází k nové volbě.

Na směrovačích Cisco se při volbě nejprve porovnává administrativní vzdálenost a následně metrika. Pokud jsou výsledky stále rovnocenné, vybere se směrovač s vyšší IP adresou.

Volba je zahájen po objevení nové RPA. Směrovač vyšle Offer zprávu obsahující jeho metriku. V případě, že přijde Offer zpráva od jiného směrovače s nižší metrikou, směrovač se již volby dále neúčastní. Cílem je, aby všechny směrovače kromě toho s nejlepší metrikou přestaly vysílat Offer zprávy. Směrovač se prohlásí za DF a vyšle všem směrovačům na lince Winner zprávu.

DF směrovač je pak zodpovědný za přeposílání multicastových paketů, které se na lince objeví, směrem k RP. Tím se zabrání, aby nebylo odesláno k RP více kopií jednoho paketu a vzniku smyček.

2.5.3 Vybudování obousměrného sdíleného stromu a jeho používání

DF směrovač zastupuje v BiDir-PM roli DR směrovače. Poté, co protokol IGMP zaregistruje nového příjemce, DF vyšle Join/Leave(*,G) zprávu směrem k RP. Pokud je směrovač, který zprávu Join/Leave příjme, DF, aktualizuje si sdílený strom stejně jako u PIM-SM, jinak zprávu zahodí.

Join/Leave zprávy se propagují ze směrovače na směrovač, dokud nedorazí na jeden ze směrovačů, který má přímo připojené RPL rozhraní. Zde zpráva končí. V případě, že RPA připřazena fyzickému směrovač (RP), končí zpráva na něm, stejně jako u PIM-SM.

Oilist obsahuje pouze rozhraní, na kterých byl směrovač zvolen jako DF a na která přišla IGMP nebo PIM Join zpráva. Větve, ve kterých se nalézají pouze zdroje a žádní příjemci, také vytváří (*,G) strom. Jejich oilist obsahuje pouze RPF rozhraní. Směrovače přímo připojené ke zdroji nemusím dělat žádné složité procedury, pouze vezmou příchozí data a zašlou ho směrem k RP.

2.5.4 Nevýhody BiDir-PIM

Již jsme uvedli, že využíváním pouze sdílených stromů bez nutnosti vytváření stromů zdrojových se šetří paměť i CPU směrovačů. Přestože došlo ke zjednodušení protokolu oproti PIM-SM, nese sebou i určité nevýhody, které je nutné zvážit před jeho nasazením.

V prvé řadě je nutné, aby všechny směrovače v PIM doméně měli BiDir mód. To se ověřuje již při objevování sousedů. PIM Hello zpráva má nově Bidirectional Capable volbu, kterou sousedy informuje, že BiDir-PIM podporuje. V případě, že by některé směrovače v doméně protokol nepodporovaly, začaly by vznikat smyčky.

To, že všechna data putují (jedním) sdíleným stromem, má za následek vytížení částí sítě kolem RPA. Přes RPL musí projít totiž veškerý provoz. Data musí dojít až na RP i v případě, že pro skupinu neexistují žádní příjemci. To je dáno tím, že BiDir-PIM nevyužívá provozní signalizaci a tudíž RP netuší, kde se nachází aktivní zdroje. U PIM-SM nejsou data na RP zasílána, pokud nejsou žádní příjemci.

BiDir-PIM je výhodné využívat v many-to-many aplikacích, kde je mnoho zdrojů i mnoho příjemců. V tomto případě je nejefektivnější. Na rozdíl od PIM-SM totiž nedochází k zvyšování zátěže úměrně s počtem zdrojů, ale zůstává stejná.

2.6 Shrnutí

V této kapitole jsme si popsali protokol PIM a jeho čtyři módy: DM, SM, SSM a BiDir. Detailně jsme se zaměřili na PIM-DM. Ten v pravidelných tříminutových intervalech zaplavuje celou síť multicastem. Segmenty, kde koncoví uživatelé o multicastová data nemají zájem, musí toto explicitně dát najevo. PIM-DM proto bude nejefektivnější v sítích, kde v každé podsíti existuje nějaký příjemce. V jiných případech se jedná o plýtvání prostředků a může být výhodnější použít častěji implementovaný protokol PIM-SM.

Protokol PIM-SM naopak nezasílá žádná data, dokud si o ně směrovače nezažádají, což celý proces zefektivňuje. Zavádí nový pojem RP směrovač a využívá sdílené stromy, jejichž kořeny se nachází v RP. Využívá ale stále i zdrojové stromy - od zdroje k RP nebo pro zkrácení cesty od zdroje k cíli. Sdílené stromy totiž nemusí být nejefektivnější při distribuci multicastu, pokud se třeba příjemce nalézá blízko zdroje.

Nejpoužívanějším protokolem je v dnešní době právě PIM-SM následovaný protokolem PIM-DM. Aby byly směrovací protokoly ještě efektivnější, vznikají jejich další verze. Za zmínku stojí PIM-SSM a BiDir-PIM. PIM-SSM umožňuje přijímat multicast od vybraných zdrojů. BiDir-PIM využívá zdroje efektivněji než PIM-SM tím, že vůbec nepracuje se zdrojovými stromy.

Kapitola 3

Podpora multicastu na Cisco zařízeních

V této kapitole se budeme zajímat o konfiguraci protokolu PIM na směrovačích. Vybraly jsme směrovače značky Cisco, neboť jsou značně rozšířené v komerční sféře a jsou také majoritně zastoupeny ve školních laboratořích. Celá problematika je velmi dobře popsaná v [8, kapitola 1].

Zaměříme se pouze na multicastové směrování pomocí protokolu PIM, které je hlavní náplní této práce. Pro praktickou funkčnost sítě je samozřejmě také nutná konfigurace protokolu IGMP. Tu naleznete v [14, kapitola 3] a [8, strana 455]. V celé kapitole budeme při konfiguraci předpokládat, že konfigurovaný směrovač má název `R`.

3.1 Základní konfigurace

V této sekci vycházíme z [8, strana 445] [6, kapitola 7] [9, kapitola 7].

Základní konfigurace skládá ze tří až čtyř kroků:

1. Globální povolení multicastového směrování.
2. Spuštění požadovaného módu (SM, DM) na potřebných rozhraních.
3. Nastavení RP směrovače (pro SM mód).
4. Volitelně State Refresh nastavení.

3.1.1 Povolení multicastového směrování

Multicastové směrování je na Cisco směrovačích implicitně vypnuté. Zapneme ho v globálním konfiguračním rozhraní:

```
R(config)# ip multicast-routing
```

3.1.2 Spuštění požadovaného módu směrování

Následně můžeme multicastové směrování spustit v jednom ze tří módů: dense (PIM-DM), sparse (PIM-SM) nebo sparse-dense. Poslední jmenovaný je vlastní Cisco mód, který umožňuje provozovat v jedné síti některé multicastové skupiny pod protokolem PIM-DM

a jiné pod protokolem PIM-SM. Který protokol se pro danou multicastovou skupinu použije, záleží na tom, zda pro skupinu existuje RP směrovač. Pokud neexistuje, využije se PIM-SM, pokud existuje, použije PIM-DM.

Další výhodou sparse-dense módu je, že Auto-RP (viz 3.2.1) zprávy mohou být distribuovány v dense módu a ostatní multicastová data mohou používat sparse mód. Mód se konfiguruje na rozhraní:

```
R(config)# interface interface-type interface-number
R(config-if)# ip pim {dense-mode | sparse-dense-mode | sparse-mode}
```

Tento příkaz také aktivuje IGMP protokol.

3.1.3 Nastavení RP směrovače

V případě, že se rozhodneme využívat sparse mód (alespoň pro jednu multicastovou skupinu), je také nutné nastavit některý směrovač v síti jako RP. Toto je možné provést dynamicky nebo staticky. Statické nastavení je na první pohled jednodušší, avšak skýtá několik nevýhod. Nastavení je potřeba udělat na všech směrovačích v PIM doméně, což může být nevýhoda, pokud provádíme konfiguraci ručně. Je to časově náročné a náchylné na chyby. Hlavní problém, ale tkví v možném výpadku RP směrovače. V tom případě by přestal fungovat celý multicastový provoz v síti.

```
R(config)# ip pim rp-address ip-address [access-list] [override]
```

V příkazu zadáme IP adresu RP směrovače. Volitelně můžeme přidat access list, který bude obsahovat IP adresy všech multicastových skupin, které pod RP patří. Klíčové slovo `override` použijeme, pokud nastavíme dynamickou i statickou volbu RP a statickou chceme upřednostnit.

3.1.4 State Refresh nastavení

Podle RFC 3973 směrovač přímo připojený ke zdroji multicastu automaticky odesílá v pravidelných intervalech State Refresh zprávy (viz 2.2.2), které zabrání periodickému zaplavování sítě multicastovými daty, jak je známo z PIMv1.

U Cisco směrovačů je ale vše jinak. Standartní chování je takové, že směrovač přijaté State Refresh zprávy zpracuje, ale sám zprávy nikdy nevysílá. Na rozhraní, kde očekáváme připojený zdroj multicastu, můžeme nastavit, aby směrovač vysílal State Refresh zprávy:

```
R(config)# interface interface-type interface-number
R(config-if)# ip pim state-refresh origination-interval [interval]
```

Nepovinným parametrem je zadání intervalu, ve kterém se budou State Refresh zprávy odesílat. Pokud nebude nastaven, zprávy se budou odesílat co 60s. Můžeme také na směrovači zcela zakázat zpracovávání těchto zpráv příkazem:

```
R(config)# ip pim state-refresh disable
```

3.1.5 Praktické příklady

Tyto příklady jsou převzaty z [6, kapitola 7]. První příklad nám ukazuje velmi jednoduché nastavení PIM-DM na jednom ethernetovém rozhraní:

```
ip multicast-routing
interface ethernet 0
   ip pim dense-mode
```

V druhém příkladu nastavujeme na rozhraní mód sparse (PIM-SM). V tomto případě je nutné taktéž nastavit IP adresu RP směrovače (10.8.0.20). Ten bude používán pouze pro multicastovou skupinu 224.0.0.0, jak je uvedeno v příslušném access listu:

```
ip multicast-routing
ip pim rp-address 10.8.0.20 1
interface ethernet 1
   ip pim sparse-mode
access-list 1 permit 224.0.0.0 15.255.255.255
```

3.2 Dynamická konfigurace RP směrovače

Implicitně Cisco směrovače používají PIM verze 2. PIM verzi je možné změnit příkazem:

```
R(config-if)# ip pim version {1 | 2}
```

3.2.1 Dynamická konfigurace RP směrovače v PIM verze 1

Standard PIM verze 1 neobsahuje dynamickou konfiguraci RP směrovačů. Cisco si proto vytvořilo svůj vlastní protokol Auto-RP, který to umožňovalo. V síti existuje mapovací agent, který sdružuje informace o RP kandidátech. Ty pak rozesílal multicastem všem směrovačům v síti. Pro nastavení směrovače jako mapovacího agenta použijeme příkaz:

`R(config)# ip pim send-rp-discovery [`*interface-type interface-number*`] scope` *ttl-value*

Rozhraní určuje IP adresu, která se bude šířit jako IP adresa mapovacího agenta. Většinou se používá lokální smyčka (loopback). TTL umožňuje omezit rozsah šíření. Na každém RP kandidátovi nastavíme:

`R(config)# ip pim send-rp-announce` {*interface-type interface- number*} `|` *ip-address* `scope` *ttl-value* `[group-list` *access-list*`]`

U tohoto příkazu je navíc možnost přidat seznam multicastových skupin, pro které se stává směrovač RP kandidátem.

3.2.2 Dynamická konfigurace RP směrovače v PIM verze 2

U PIM verze 2 je podobný mechanismus jako Auto-RP již součástí protokolu. Nazývá se Bootstrap Router (viz 2.3.4) a je ho vhodné použít zejména tehdy, pokud se v síti nacházejí i směrovače jiných výrobců. Podobně jako u Auto-RP musí existovat v síti jeden směrovač označován jako BSR, který bude zaplavovat síť seznamem RP směrovačů, a RP kandidáti, jejichž IP adresy se budou v seznamu nacházet.

BSR kandidáta nastavíme příkazem:

`R(config)# ip pim bsr-candidate` *interface-type interface-number hash-mask-length* [*priority*]

Taktéž zadáme rozhraní, jehož IP adresa bude šířena sítí. Maska (max. 32 bitů) slouží pro přiřazení RP k multicastovým skupinám. Priorita je číslo od 0 do 255 (implicitně 0), které slouží při výběru mezi BSR kandidáty. BSR se stává směrovač s nejvyšším číslem. Ostatní slouží jako záložní BSR. RP kandidát se nastaví příkazem:

`R(config)# ip pim rp-candidate` *interface-type interface-number ttl* `[group-list` *access-list-number*]

Ve větších sítích je také vhodné označit hranice PIM domény. Vybereme směrovače, které budou hraničními, a v konfiguračním módu rozhraní zadáme příkaz:

```
R(config-if)# ip pim border
```

Nebo novější příkaz:

```
R(config-if)# ip pim bsr-border
```

Stejně tak můžeme zabránit, aby multicast zvenčí byl zasílán dovnitř naší sítě. Vhodné je zakázat Auto-RP zprávy.

`R(config-if)# ip multicast boundary` *access-list-number*

Access list bude obsahovat zakázané multicastové adresy. Pro Auto-RP jsou to adresy 224.0.1.39 a 224.0.1.40.

3.2.3 Praktické příklady

Tyto příklady jsou převzaty z [6, kapitola 7]. První příklad ukazuje konfiguraci BSR směrovače. Má dvě ethernetové rozhraní ve sparse-dense módu. Unicastové směrování provádí protokol OSPF. Odesílatelem zprávy o BSR kandidatuře je rozhraní Ethernet 1, maska je 30 a priorita 10. Směrovač je také RP kandidát pro multicastové skupiny definované v access listu 5, tedy 239.255.2.0/24.

```
ip multicast-routing
!
interface Ethernet0
 ip address 172.21.24.18 255.255.255.248
 ip pim sparse-dense-mode
!
```

```
interface Ethernet1
 ip address 172.21.24.12 255.255.255.248
 ip pim sparse-dense-mode
!
router ospf 1
 network 172.21.24.8 0.0.0.7 area 1
 network 172.21.24.16 0.0.0.7 area 1
!
ip pim bsr-candidate Ethernet1 30 10
ip pim rp-candidate Ethernet1 group-list 5
access-list 5 permit 239.255.2.0 0.0.0.255
```

Druhý příklad představuje nastavení hraničního směrovače. Hranice se nachází na ethernetovém rozhraní 1. Kromě toho jsou také zahazovány pakety z multicastových skupin uvedených v access list 1, které přijdou do PIM domény zvenčí. Podle multicastových adres vidíme, že jsou zahazovány lokální multicastové IP adresy (239.0.0.0/8) a IP adresy Auto-RP.

```
ip multicast-routing
!
interface Ethernet1
 ip address 172.21.24.18 255.255.255.248
 ip pim sparse-dense-mode
 ip pim border
 ip multicast boundary 1
!
access-list 1 deny 239.0.0.0 0.255.255.255
access-list 1 deny 224.0.1.39 0.255.255.255
access-list 1 deny 224.0.1.40 0.255.255.255
access-list 1 permit 224.0.0.0 15.255.255.255
```

3.3 Monitorování a verifikace

Pro kontrolu správné funkčnosti multicastového směrování existuje několik `show` příkazů.

`R# show ip mroute [`*group-address*`] [summary] [count][active` *kpbs*`]`

Příkazem zobrazíme multicastovou směrovací tabulku. Můžeme zadat konkrétní multicastovou skupinu, pro kterou ji chceme zobrazit. Zkrácený výpis získáme přepínačem `summary`, `count` nám vrátí statistky a `active` informace pouze o aktivních multicastových skupinách. Výpis obsahuje multicastové stromy: zdrojové jsou značeny (IP_adresa_zdroje, multicastová_IP_adresa) a sdílené (*, multicastová_IP_adresa). U každého stromu je popsáno příchozí a odchozí rozhraní.

```
R# show ip mroute 225.25.25.25
IP Multicast Routing Table
Flags: D - Dense, S - Sparse, B - Bidir Group, s - SSM Group,
```

```
       L - Local, P - Pruned, R - RP-bit set, F - Register flag,
       T - SPT-bit set, J - Join SPT, C - Connected
 Timers: Uptime/Expires
 Interface state: Interface, Next-Hop or VCD, State/Mode

(*, 225.25.25.25), 02:39:52/00:03:20, RP 10.100.1.1, flags: SJCL
  Incoming interface: Null, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Sparse-Dense, 00:18:52/00:03:20
    Loopback1, Forward/Sparse-Dense, 02:39:52/00:02:15

(10.100.20.4, 225.25.25.25), 00:03:14/00:02:59, flags: LT
  Incoming interface: FastEthernet0/0, RPF nbr 10.100.13.3
  Outgoing interface list:
    Loopback1, Forward/Sparse-Dense, 00:03:15/00:02:14
```

Pro zjištění PIM sousedů použijeme jeden z následujících příkazů:

```
R# show ip pim interface [detail]
R# show ip pim neighbor
```

Výpis obsahuje téměř totožné informace. Vždy je to adresa souseda, rozhraní, verze protokolu PIM, priorita a mód.

```
R# show ip pim neighbor
PIM Neighbor Table
Mode: B - Bidir Capable, DR - Designated Router, N - Default DR Priority,
      S - State Refresh Capable
Neighbor          Interface            Uptime/Expires    Ver   DR
Address                                                        Prio/Mode
172.16.13.3       FastEthernet0/0      00:07:22/00:01:19 v2    1 / DR S
172.16.102.2      Serial0/0/0          00:07:23/00:01:22 v2    1 / S
172.16.103.3      Serial0/0/1          00:07:23/00:01:29 v2    1 / S
```

Nakonfigurované RP směrovače prozkoumáme příkazem:

```
R# show ip pim rp [group-name | group-address | mapping]
```

Bez zadání přepínače získáme výpis RP pro aktivní multicastové skupiny. Přepínač mapping vypíše pro každou IP adresu skupiny RP, případně můžeme získat pouze adresu RP pro zadanou skupinu.

```
R# show ip pim rp
Group: 226.26.26.26, RP: 10.100.3.3, v2, v1, uptime 00:53:51, expires...
Group: 225.25.25.25, RP: 10.100.1.1, v2, v1, next RP-reachable in...
```

Podobně můžeme získat informace o BSR směrovači pomocí příkazu:

```
R# show ip pim bsr
```

Příkazem:

```
R# show ip pim rp-hash {group-address | group-name}
```

zobrazíme, které RP bylo vybráno pro zadanou multicastovou skupinu.

```
R# show ip pim rp-hash 239.1.1.1
RP 172.21.24.12, v2
   Info source: 172.21.24.12, via bootstrap
      Uptime: 05:15:33, expires: 00:02:01
```

Následující příkaz nám zobrazí informace o RPF:

```
R# show ip rpf 172.16.20.4
RPF information for ? (172.16.20.4)
RPF interface: FastEthernet0/0
RPF neighbor: ? (172.16.13.3)
RPF route/mask: 172.16.20.0/24
RPF type: unicast (eigrp 1)
RPF recursion count: 0
Doing distance-preferred lookups across tables
```

Můžeme také použít nástroj `mrinfo`, který vypíše všechny multicastové sousedy:

```
R# mrinfo
172.16.13.1 [version  12.4] [flags: PMA]:
192.168.1.1 -> 0.0.0.0 [1/0/pim/querier/leaf]
172.16.13.1 -> 172.16.13.3 [1/0/pim]
172.16.102.1 -> 172.16.102.2 [1/0/pim]
172.16.103.1 -> 172.16.103.3 [1/0/pim]
```

Statistiky o multicastovém provozu procházejícím směrovačem získáme příkazem:

```
R# show ip multicast interface
```

Pro každé rozhraní na směrovači kromě IP adresy vypíše, zda je na něm povoleno multicastové směrování, typ multicastového přepínání a počet paketů, které přes rozhraní prošly.

```
R1# show ip multicast interface
FastEthernet0/0 is up, line protocol is up
  Internet address is 172.16.13.1/24
  Multicast routing: enabled
  Multicast switching: fast
```

```
Multicast packets in/out: 524/6
Multicast TTL threshold: 0
Multicast Tagswitching: disabled
```

V neposlední řadě také můžeme využít debugovací výpisy. Zapneme je příkazem:

```
R# debug ip pim
```

3.4 Konfigurace PIM-SSM a BiDir-PIM

Na závěr se zběžně podíváme na konfiguraci méně rozšířených protokolů PIM-SSM [8, strana 505] a BiDir-PIM [8, strana 517]. Při znalosti konfigurace PIM-DM a PIM-SM, je konfigurace těchto protokolů již velmi snadná, neboť využívají stejných postupů, které jsou jen mírně modifikovány. Pro všechny protokoly PIM je nutné na začátku povolit směrování multicastu (viz 3.1.1).

3.4.1 Konfigurace PIM-SSM

PIM-SSM se globálně nastavuje příkazem:

```
R(config)# ip pim ssm [default | range access-list]
```

Součástí příkazu je možné definovat rozsah multicastových adres pro PIM-SSM. Použitím klíčového slova `default` se rozsah nastaví na 232/8. Můžeme si také rozsah definovat sami pomocí access listu a využít klíčové slovo `range`.

Abychom povolili PIM na jednotlivých rozhraních, použijeme stejný příkaz jako u PIM-SM. Mód může být buď sparse, nebo sparse-dense:

```
R(config-if)# pim {sparse-mode | sparse-dense-mode}
```

Tentokrát je vhodné upozornit i na konfiguraci IGMP protokolu. PIM-SSM může pracovat pouze s IGMPv3. Implicitně využívají Cisco zařízení IGMPv2. Cisco zařízení také nabízejí vlastní protokoly IGMPv3lite a URD, které předcházely IGMPv3. Při konfiguraci IGMP můžeme proto využít jeden z těchto tří příkazů:

```
R(config-if)# ip igmp version 3
R(config-if)# ip igmp v3lite
R(config-if)# ip urd
```

Možnosti monitorování a verifikace jsou totožné s těmi, které jsme si uvedli v sekci 3.3. Příklad jednoduché konfigurace PIM-SSM s IGMPv3 může vypadat takto:

```
ip multicast-routing
!
interface Ethernet3/1
 ip address 172.21.200.203 255.255.255.0
 description backbone interface
 ip pim sparse-dense-mode
```

```
!
interface Ethernet3/2
 ip address 131.108.1.2 255.255.255.0
 ip pim sparse-dense-mode
 description ethernet connected to hosts
 ip igmp version 3
!
ip pim ssm default
```

3.4.2 Konfigurace BiDir-PIM

Konfigurace BiDir-PIM je taktéž velmi jednoduchá. Nejprve povolíme BiDir-PIM globálně:

```
R# ip pim bidir-enable
```

Následně je nutné nastavit RP směrovače. Postup je totožný s PIM-SM. RP můžeme nastavit staticky (3.1.3), dynamicky pomocí Auto-RP (3.2.1) či Bootstrap Router mechanismu (3.2.2). Na konec všech příkazů je nutné přidat klíčové slovo `bidir`:

```
R(config)# ip pim rp-address ip-address [access-list] [override] bidir
```

`R(config)# ip pim send-rp-announce` {*interface-type interface- number* | *ip-address*} `scope` *ttl-value* `[group-list` *access-list*`] bidir`

`R(config)# ip pim rp-candidate` *interface-type interface-number ttl* `[group-list` *access-list-number*`] bidir`

Možnosti monitorování a verifikace jsou totožné s těmi, které jsme si uvedli v sekci 3.3. Navíc můžeme ještě použít příkaz:

```
R# show ip pim interface df
```

Příkaz zobrazí informace o zvoleném DF směrovači pro každé RP na rozhraní a metriku, která se s daným DF pojí.

Nakonec uvádíme příklad jednoduché konfigurace BiDir-PIM směrovače:

```
ip multicast-routing
ip pim bidir-enable
!
interface loopback 0
 description One Loopback adddress for this routers Bidir Mode RP
 ip address 10.0.1.1 255.255.255.0
 ip pim sparse-dense-mode
!
interface loopback 1
 description One Loopback adddress for this routers Sparse Mode RP
 ip address 10.0.2.1 255.255.255.0
 ip pim sparse-dense-mode

ip pim send-rp-announce Loopback0 scope 10 group-list 45 bidir
```

```
ip pim send-rp-announce Loopback1 scope 10 group-list 46
ip pim send-rp-discovery scope 10

access-list 45 permit 224.0.0.0 0.255.255.255
access-list 45 permit 227.0.0.0 0.255.255.255
access-list 45 deny   225.0.0.0 0.255.255.255
access-list 46 permit 226.0.0.0 0.255.255.255
```

Multicastové skupiny 224/8 a 227/8 využívají BiDir-PIM, skupina 226/8 PIM-SM a 225/8 využívá PIM-SM pro své šíření.

3.5 Shrnutí

V této kapitole jsme si shrnuli, jak je možné na Cisco směrovačích konfigurovat PIM protokoly. Podrobně jsme si předvedli konfiguraci PIM-DM a PIM-SM. Konfiguraci RP směrovačů můžeme provádět staticky nebo dynamicky využitím Cisco protokolu Auto-RP či Bootstrap Router mechanizmu, který je součástí protokolu PIM od jeho verze 2.

Ukázali jsme si mnoho příkazů, které můžeme používat ke kontrole správnosti chodu multicastového směrování a případným opravám. Na závěr jsme také nakoukly pod pokličku konfigurace protokolů PIM-SSM a BiDir-PIM, které se příliš neliší od předchozích dvou protokolů. Všechny konfigurace jsme demonstrovali na jednoduchých příkladech.

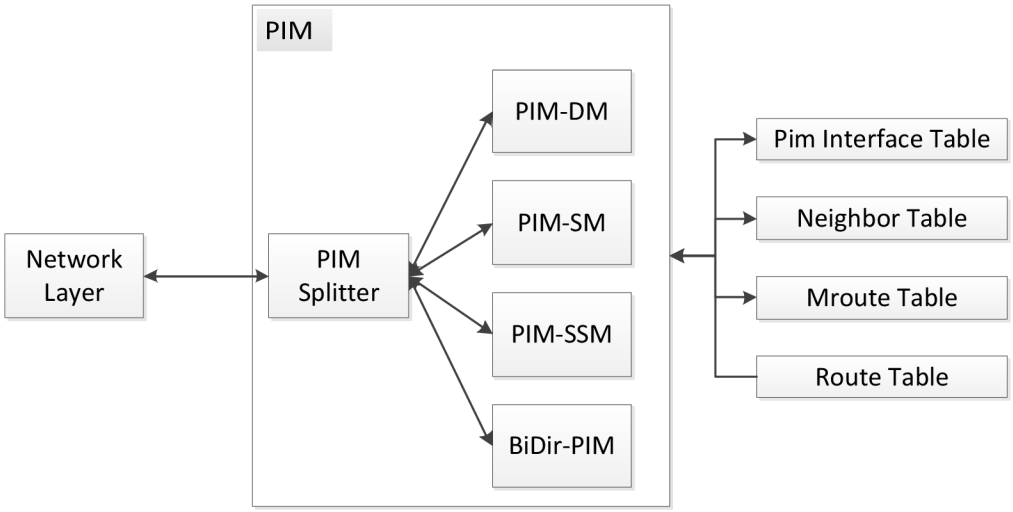
Kapitola 4

Návrh rozšíření simulátoru o podporu multicastového směrování

Nyní již máme dobré teoretické základy o směrovacích protokolech PIM (kapitola 2). Víme, jak se konfigurují v praxi (kapitola 3), a seznámili jsme se se simulačním nástrojem (kapitole 1), který budeme při této práci používat. S těmito poznatky můžeme vytvořit návrh rozšíření simulátoru OMNeT++ o podporu multicastového směrovacího protokolu PIM.

4.1 Návrh architektury

PIM je protokol síťové vrstvy, proto navážeme PIM model na model síťové vrstvy (NetworkLayer) podobně, jako je tomu u existujícího modelu protokolu OSPF. Model NetworkLayer bude nutné upravit tak, aby pakety označené protokolovým číslem 103 zasílal do PIM modelu.



Obrázek 4.1: Návrh PIM modelu.

Většina směrovacích protokolů, které již byly do OMNeT++ implementovány, jsou jed-

noduché modely. Protokol PIM je ale v tomto ohledu složitější. Jak jsme si uvedli v kapitole 2, protokol PIM může pracovat v několika módech. Jejich logika postavena na konečných automatech se mnohdy výrazně liší, a proto bude nutné pro protokol vytvořit složený model.

Uvnitř modelu bude jednoduchý model pro každý PIM mód. Na síťové úrovni ale není možné jednotlivé módy od sebe odlišit. Všechny mají stejné protokolové číslo. Jediný způsob, jak je odlišit, je vyčíst tuto informaci z konfiguračního souboru. Z tohoto důvodu budeme potřebovat ještě jeden jednoduchý model, PIM Splitter, který bude příchozí zprávy rozesílat do modelů příslušných PIM módu.

Mimo uvedené musíme vytvořit i tabulky, do kterých se budou zapisovat zásadní informace pro protokol. Je to již zmíněná tabulka PIM rozhraní, tabulka sousednosti a směrovací multicastová tabulka. Ty budou korespondovat s Cisco tabulkami, které je možné vyvolat příkazy `show ip pim interface`, `show ip pim neighbor` a `show ip mroute` (viz 3.3). Akce nad těmito tabulkami a jiné úkony, které jsou pro všechny módy společné, mohou být spravovány PIM Splitterem. Návrh modelu je na obrázku 4.1.

4.2 Abstraktní datové typy

Modul PIM vyžaduje nové abstraktní typy, které se v OMNeT++ zatím nevyskytují. Z tabulek, které potřebuje protokol PIM pro svou práci (viz obrázek 4.1) je vyhovující pouze implementace unicastové směrovací tabulky.

4.2.1 Tabulka PIM rozhraní

Tabulka PIM rozhraní bude uchovávat informace o rozhraních, na kterých je zapnut protokol PIM. Tyto informace budou načteny z konfiguračního souboru a budou sloužit především pro rozesílání PIM zpráv. Pro každé rozhraní bude tabulka uchovávat jeho identifikátor a mód protokolu PIM, který je na rozhraní povolen.

4.2.2 Tabulka sousednosti

Tabulka sousednosti bude uchovávat informace o přímo připojených směrovačích, které také mají na rozhraní zapnutý protokol PIM. Tabulka bude vytvářena a měněna na základě zpráv PIM Hello a časovače Neighbor Liveness Timer (NLT).

Ukládat se bude IP adresa souseda, rozhraní, na které je soused napojen, verze protokolu PIM a Neighbor Liveness Timer, po jehož vypršení dojde k odstranění souseda z tabulky.

4.2.3 Multicastová směrovací tabulka

Jak jsme si uvedli v části 1.3.1, knihovna INET má implementovanou směrovací tabulku i pro multicast. V třídě `RoutingTable` se nachází struktura pro multicastovou cestu. Ta ale zaznamenává pouze multicastovou IP adresu a výstupní rozhraní.

Tento způsob je vhodný pro statické multicastové adresy jako jsou IP adresy používané směrovacími protokoly. V případě obecného multicastového provozu potřebujeme uchovávat v tabulce více informací, proto si vytvoříme vlastní multicastovou tabulku.

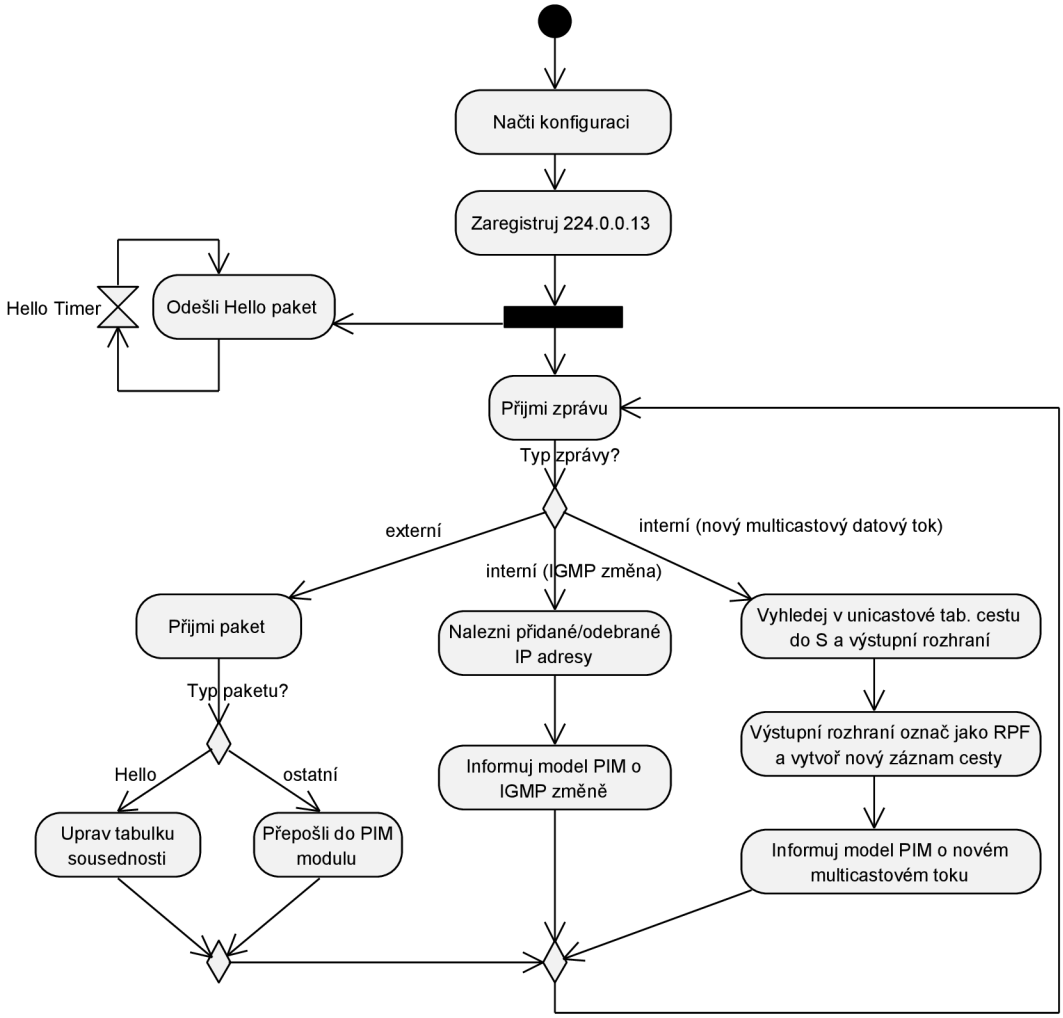
V tabulce budeme uchovávat označení stromu (*,G) nebo (S,G) s příslušnými IP adresami, případně adresu RP, časovače, příchozí a odchozí rozhraní. U příchozího rozhraní musíme znát název rozhraní a také IP adresu RPF souseda. Odchozí rozhraní bude tvořeno seznamem obsahující název rozhraní, mód, stav a příslušné časovače.

K multicastové tabulce nebude přistupovat pouze PIM modul a tudíž ji bude nutné implementovat jako třídu. Kromě datové struktury popsané výše bude obsahovat i potřebné funkce pro práci s tabulkou, např. vložení nového záznamu, odstranění záznamu, editace záznamu.

4.3 PIM Splitter

PIM Splitter provádí obecné operace, které mohou využít všechny PIM módy bez rozdílů. PIM zprávy jsou nejprve směrovány na něj a on rozhodne, co se s nimi dále bude dít.

Základní funkcionalitu PIM Splitteru můžete vidět na diagramu aktivit (obrázek 4.2). PIM Splitter nejprve načte PIM konfiguraci z konfiguračního souboru, kde se dozví především to, jaký PIM mód je na směrovači nastaven a na kterých rozhraních.



Obrázek 4.2: Diagram aktivity pro PIM Splitter.

PIM si musí zaregistrovat IP adresu 224.0.0.13, která označuje všechny PIM směrovače. Tato multicastová adresa se používá pro zasílání většiny PIM zpráv. Po základní inicializaci PIM Splitter vyčkává na jednu z mnoha akcí, které mohou nastat. Diagram zachycuje jen

ty, které jsou zásadní.

Základní funkcionalitou PIM Splitteru je rozesílání přijatých PIM zpráv do správných PIM modulů podle konfigurace. PIM Hello zprávy ale může zpracovávat sám. Podle jejich obsahu upraví tabulku sousednosti (viz 4.2.2). Všechny ostatní typy zpráv přepošle k dalšímu zpracování.

PIM Splitter musí periodicky po uplynutí časovače Hello vysílat PIM Hello zprávy na všechny rozhraní, na kterých je nastavený PIM. Směrovač si tak udržuje sousedství s ostatními PIM směrovači v síti.

V případě, že přijdou na směrovač data z nového multicastového toku, musí se zapsat do multicastové směrovací tabulky (viz 4.2.3). PIM Splitter nejprve vyhledá v unicastové směrovací tabulce cestu ke zdroji multicastu S. Výstupní rozhraní uvedené v tabulce se označí jako RPF rozhraní. Nakonec se odešle interní zpráva do příslušného PIM modulu, aby do nového záznamu doplnil odchozí rozhraní (oilist).

PIM Splitter musí také sledovat změny multicastových adres na rozhraní. Pokud k nějaké změně dojde, zjistí, které IP adresy byly přidány/odebrány, a odešle interní zprávu PIM modulu, aby na změnu zareagoval.

Všechny uvedené akce se mohou opakovat do nekonečna.

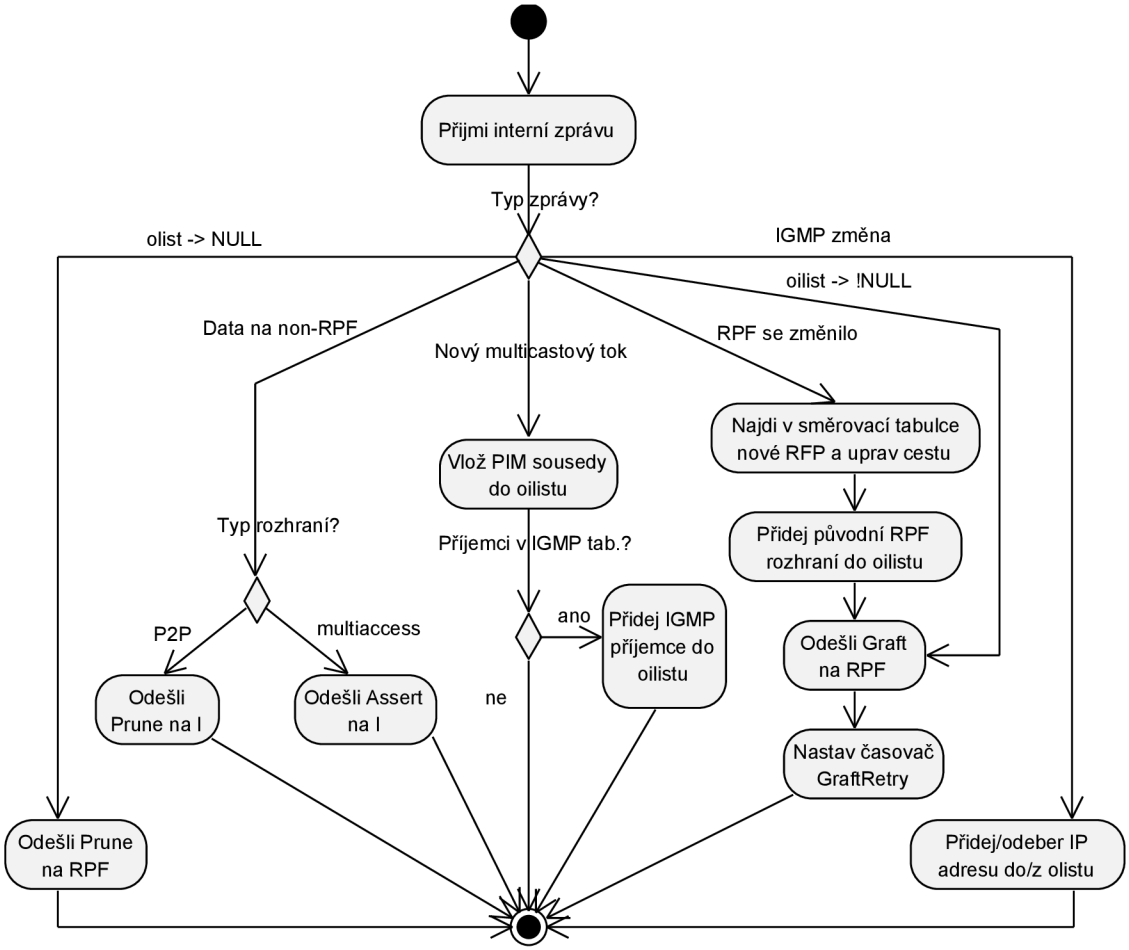
4.4 PIM-DM

Přestože funkcionalita protokolu PIM-DM je v RFC 3973 (viz sekce 2.2) popsána pomocí poměrně složitých stavových automatů, sami autoři RFC nedoporučují implementaci protokolu pomocí nich. Z tohoto důvodu jsme se snažili celou problematiku zjednodušit.

Obrázek 4.3: Základní diagram aktivity pro PIM-DM modul

V rámci návrhu modulu byl vytvořen diagram aktivit, který jsme pro lepší přehlednost rozdělili do tří částí. Základní činnost je naznačena na diagramu 4.3. PIM-DM v cyklu přijímá zprávy, které do modulu přicházejí. Ty jsou buď interního charakteru (diagram 4.4), nebo se jedná o zprávy od jiných PIM směrovačů v doméně (diagram 4.5).

Mezi interní zprávy (diagram 4.4) patří informace o novém multicastovém toku, vyprázdnění oilistu, znovunaplnění oilistu, změně RPF rozhraní, IGMP změně (přihlášení/odhlášení příjemce) a multicastových datech, které přišly na non-RPF rozhraní[1].



Obrázek 4.4: Diagram aktivity pro příjem interních zpráv modulem PIM-DM.

V případě, že na směrovač dorazil zcela nový multicastový tok, PIM Splitter již doplnil informaci o vstupním rozhraní do záznamu, který PIM-DM obdržel. Chybí ale informace o odchozích rozhraních (oilist). Modul naplní oilist rozhraními ke všem PIM sousedům z tabulky sousednosti (kromě RPF rozhraní). Pak se podívá, jestli pro danou skupinu existují nějací koncoví příjemci. Pokud ano, přidá do oilistu i rozhraní, ke kterým jsou připojení.

Pokud je oilist prázdný, může směrovač zastavit příjem multicastu vysláním zprávy Prune na RPF rozhraní. Pakliže se původně prázdný oilist naplní, vyšle na RPF Graft zprávu, kterou dá najevo, že má o multicast znovu zájem. Stejně postupuje, když se RPF rozhraní změní a oilist není prázdný. Navíc ale musí přidat původní RPF rozhraní do oilistu a naopak nové RPF rozhraní ze seznamu odebrat.

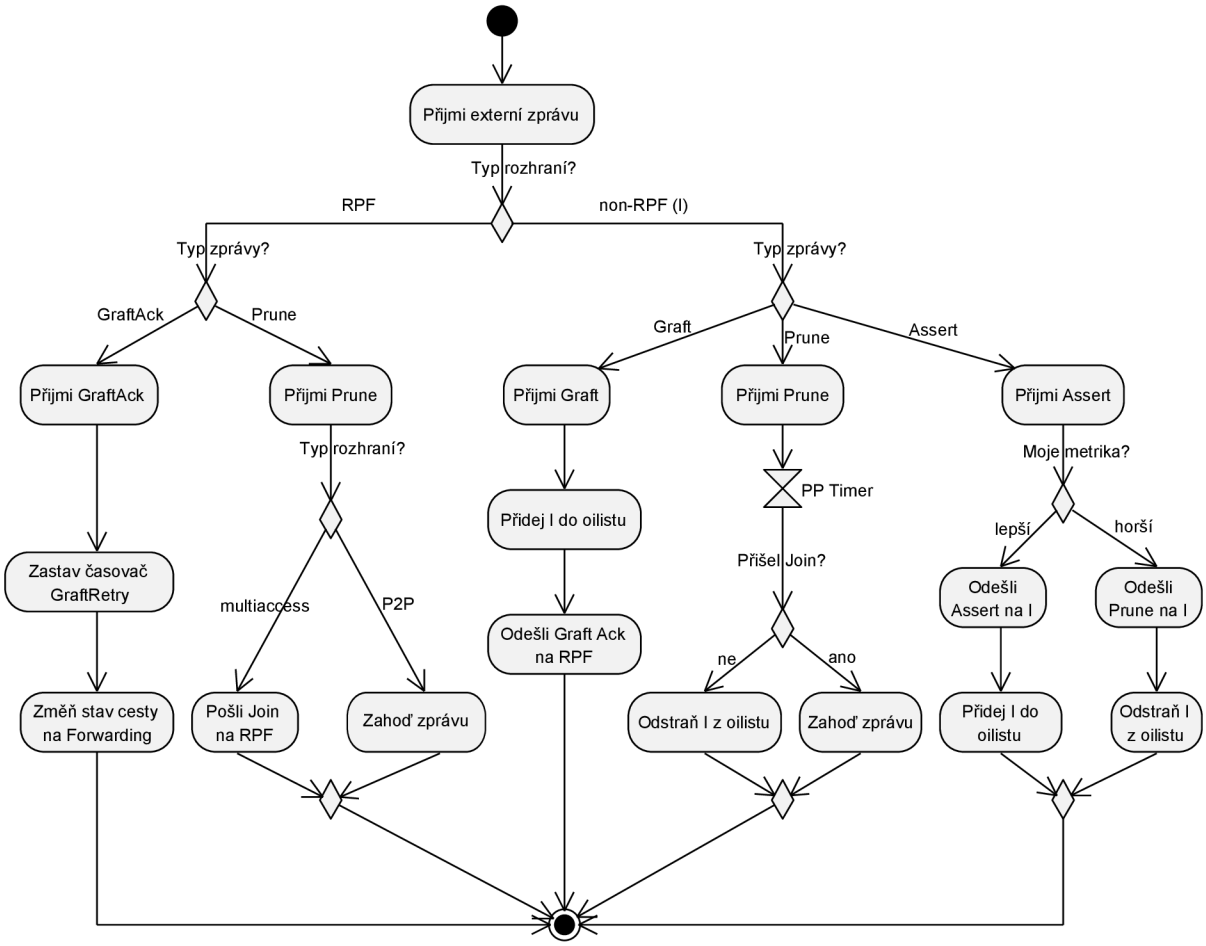
Může se stát, že k jedné LAN síti je připojeno více PIM směrovačů. To se projeví tak, že multicastová data přijdou na jiné než RPF rozhraní. Směrovač zareaguje vysláním Assert

[1]Non-RPF rozhraní je jiné PIM rozhraní než to, které bylo označeno jako RPF.

zprávy a dojde k Assert volbě (viz 2.2.2). Podobná situace může nastat na P2P lince, tehdy směrovač odešle na rozhraní zprávu Prune. Zamezí tak cyklení multicastových dat.

Pokud modul IGMP dostane informaci od přímo připojeného uživatele, že má zájem o odběr multicastových dat, nebo naopak že o multicast již nemá zájem, musí být informován i PIM-DM modul. Od PIM Splitteru dostane seznam adres, které byly přidány na rozhraní, nebo byly z rozhraní odebrány. Modul pak tyto rozhraní přidá do oilistu, nebo je z něj odebere.

U příjmu PIM zpráv (diagram 4.5) rozlišujeme, zda přišly na RPF rozhraní (od upstream směrovače) nebo na non-RPF (od downstream směrovače). V případě RPF rozhraní je zásadní příjem Prune zprávy. Taková situace nastane, pokud je k LAN připojeno více směrovačů a jeden z nich nechce dále přijímat multicast. Pokud o něj ale jiný směrovač na segmentu zájem stále má (oilist je neprázdný), musí zareagovat odesláním Join zprávy na RPF rozhraní.



Obrázek 4.5: Diagram aktivity pro příjem externích zpráv modulem PIM-DM.

Přijetím zprávy Graft-Ack na RPF rozhraní je potvrzeno, že upstream směrovač přijal zprávu Graft. Směrovač zareaguje změnou stavu cesty na Forwarding a zastavením časovače GraftRetry.

V případě, že na non-RPF rozhraní přijde zpráva Prune, znamená to, že downstream

směrovač již o multicast nemá zájem a chce se odřezat od stromu. Směrovač počká určitou dobu (PPT časovač), jestli jiný směrovač na rozhraní nepřehlasuje původní Prune zprávu zprávou Join. Pokud ne, odstraní rozhraní z oilistu. Jinak zprávu ignoruje.

Příjmem Graft zprávy na non-RPF rozhraní dává downstream směrovač najevo, že má znovu zájem o příjem multicastu a chce se připojit ke stromu. Směrovač si přidá toto rozhraní do oilistu a zašle potvrzovací zprávu Graft Ack.

Poslední zpráva Assert se týká volby směrovače, který bude zasílat data na LAN síť. Přijme-li směrovač takovou zprávu, podívá se nejprve na metriku v ní obsaženou. Pokud je horší než metrika směrovače, odešle na rozhraní svou zprávu Assert. Pokud rozhraní není v oilistu, přidá ho tam. Je-li metrika lepší, odešle na rozhraní zprávu Prune a rozhraní z oilistu odstraní.

Všechny uvedené akce se mohou opakovat do nekonečna. Diagram je značně zjednodušen, protože svoji roli bude hrát i velké množství časovačů, které PIM-DM používá (viz 2.2.1).

4.5 Shrnutí

Cílem této práce je rozšíření simulátoru OMNeT++ o směrovací protokol PIM. Tato kapitola nás k cíli výrazně přiblížila. Navrhli jsme, jak toto rozšíření provedeme. Základem je návrh struktury modelu PIM a jeho napojení na existující model směrovače v OMNeT++. Model je složený z PIM Splitteru a modelů jednotlivých PIM módu. Napojen bude na model síťové vrstvy.

OMNeT++ musíme rozšířit nejen o funkcionalitu samotného směrovacího protokolu, ale bude nutné implementovat i chybějící abstraktní datové typy, zejména tabulku sousednosti, tabulku rozhraní a multicastovou směrovací tabulku. Co se týče protokolu PIM, rozhodli jsme se v rámci této práce implementovat model protokolu PIM-DM a nezbytného PIM Splitteru. Návrh obou modelů jsme si názorně ukázali pomocí diagramu aktivit.

Kapitola 5

Konfigurace multicastu na síťových prvcích v OMNeT++

V sekci 4.3 jsme si uvedli, že na začátku simulace musí PIM Splitter načíst konfiguraci protokolu PIM. Konfigurační soubor je podobný konfiguraci reálných směrovačů, proto jsme také při jeho vytváření vycházeli z příkazů zmiňovaných v kapitole 3.

5.1 Návrh konfiguračního souboru pro protokol PIM

Konfigurační soubor je zapsán v značkovacím jazyce XML. Bližší informace ke konfiguračním souborům, jejichž struktura vznikla v rámci projektu ANSA, je možné získat v [18, kapitola 4]. My budeme přidávat konfiguraci do části ohraničenou elementy `Routing` a `Interfaces`.

V sekci `Routing` jsme vytvořili nový zanořený element `Multicast`. Pomocí parametru `enable` nastaveném na 1 povolíme multicast na směrovači. Přestože se zabýváme implementací módu dense, navrhli jsme konfigurační soubor i pro případnou implementaci sparse módu. V zanořeném elementu `Pim` můžeme nakonfigurovat informace o RP, a to buď staticky (element `RPAddress`), nebo dynamicky použitím BSR mechanismu (elementy `BSRCandidate` a `RPCandidate`).

```
<Routing>
   <Multicast enable="1">
      <Pim>
         <RPAddress>
            <IPAddress>192.168.1.2</IPAddress>
            <Acl>1</Acl>
         </RPAddress>
         <BSRCandidate>
            <Interface>eth0</Interface>
            <Mask>30</Mask>
         </BSRCandidate>
         <RPCandidate>
            <Interface>eth1</Interface>
            <Ttl>2</Ttl>
            <GroupList>1</GroupList>
         </RPCandidate>
```

```
      </Pim>
   </Multicast>
</Routing>
```

Poslední úpravy se týkají rozhraní označeného elementem `Interface`. Zde jsme zavedli nový element `Pim`, který umožňuje výběr PIM režimu v zanořeném elementu `Mode`. Mohou se zde nacházet řetězce `dense-mode` pro PIM-DM, `sparse-mode` pro PIM-SM, případně další potřebné režimy.

Prázdný nepovinný element `Border` označuje, že se jedná o PIM Multicast Border Router na rozhraní dvou PIM domén. Dalším nepovinným elementem je `StateRefresh` se zanořeným elementem `OriginationInterval`, který zapne generování PIM State Refresh zpráv. Nepovinně může obsahovat časový interval mezi odesíláním zpráv.

```
<Interfaces>
  <Interface name="eth0">
     <Pim>
        <Mode>dense-mode</Mode>
        <Border />
        <StateRefresh>
           <OriginationInterval></OriginationInterval>
        </StateRefresh>
     </Pim>
  </Interface>
</Interfaces>
```

Konfigurace protokolu PIM-DM na směrovači je velmi jednoduchá:

```
<Routing>
   <Multicast enable="1"></Multicast>
</Routing>
<Interfaces>
   <Interface name="eth0">
      <Pim>
         <Mode>dense-mode</Mode>
      </Pim>
   </Interface>
</Interfaces>
```

5.2 Načtení konfiguračních souborů

Pro načtení konfiguračního souboru z předešlé sekce 5.1) je nutné z něj vyparsovat informace, které jsou pro protokol PIM relevantní, ve formě, se kterou budou moci moduly dále pracovat.

K parsování je vhodné využít třídu `cXMLElement`, která je nedílnou součástí samotného OMNeT++. Tato třída obsahuje metody pro přístup k jednotlivým uzlům a elementům XML struktury jako jsou `getElementByPath()`, `getNodeValue()`, atd.

Původní myšlenka byla taková, že konfiguraci bude načítat PIM Splitter. Může provést načtení konfigurace bez ohledu na to, jaký režim protokolu PIM se bude používat. Vytvořili jsme metodu `LoadConfigFromXML()`, která je volána při inicializaci modulu. Metoda načte konfigurační soubor a pomocí metod třídy `cXMLElement` najde v souboru konfiguraci pro protokol PIM. Nalezené informace ukládá do tabulky PIM rozhraní.

Tento přístup ale není vhodný pro další rozšiřování a není dostatečně obecný. Rozhodli jsme se proto navázat na práci Marka Černého. Ten v rámci své diplomové práce[10] vytvořil modul `deviceConfigurator`, který by měl při inicializaci simulace načíst konfigurační soubory centrálně. Zatím umí načítat konfiguraci pouze pro moduly podporující IPv6.

Provedli jsme rozšíření třídy `deviceConfigurator` o dvě nové metody. Metoda `isMulticastEnabled()` nalezne element Multicast a zjistí, jestli hodnota jeho atributu `enabled` je nastavena na 1. V případě, že ano, multicast je na zařízení povolen a může se zavolat metoda `loadPimInterfaceConfig()`, která načte konfiguraci pro protokol PIM.

Metoda se volá pro každé rozhraní nalezené v konfiguračním souboru. Kromě toho, že se z nalezených příkazů vytvoří nový záznam do tabulky PIM rozhraní, metoda také přidá multicastovou adresu 224.0.0.13 na rozhraní.

5.3 Shrnutí

Pro vlastní simulaci je zásadní správná konfigurace síťových prvků. V této kapitole jsme si uvedli návrh rozšíření XML struktury konfiguračního souboru o konfiguraci protokolu PIM. Následně jsme si vysvětlili, jak je možné tento konfigurační soubor načíst. Můžeme k tomuto účelu využít buď metodu `LoadConfigFromXML()` přímo v třídě implementující modul `pimSplitter` nebo centralizované načítání konfigurací modulem `deviceConfigurator`, který jsme obohatili o potřebné metody. Druhý způsob je preferovanější, proto se aktuálně metoda `LoadConfigFromXML()` nevyužívá.

Kapitola 6

Implementace podpory multicastového směrování v OMNeT++

Na následujících řádcích si podrobně popíšeme, jaké C++ třídy byly vytvořeny v rámci implementace a jaké úkony provádí nejdůležitější metody. Implementace byla provedena na základě návrhu popsaného v kapitole 4.

6.1 Implementace rozšíření pro základní podporu multicastu

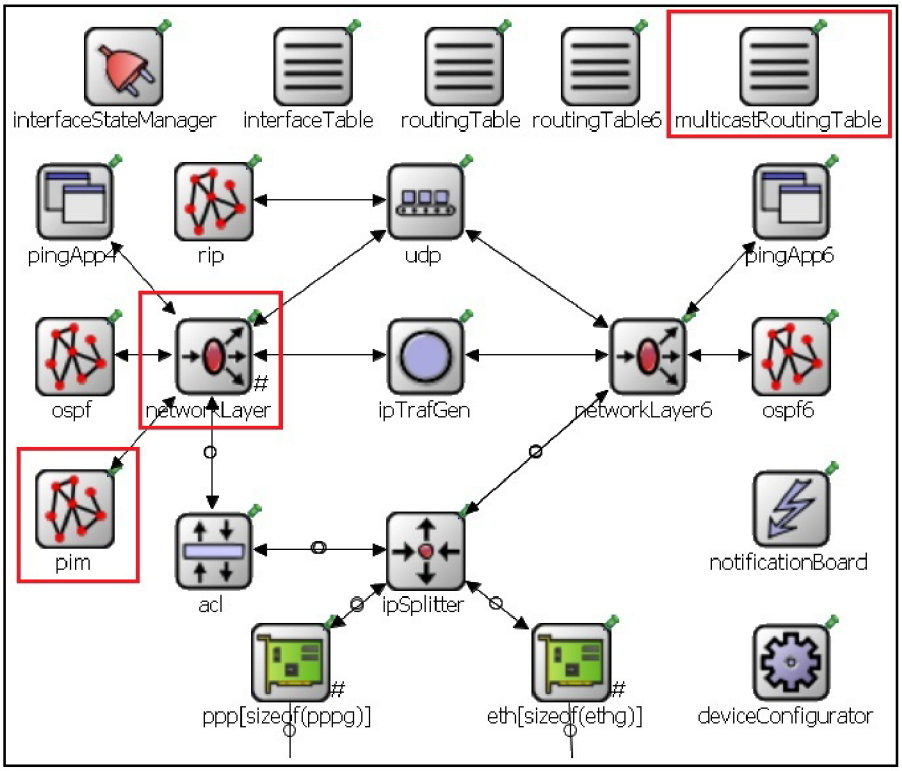
Pro podporu multicastového provozu jako takového a pro podporu dynamického multicastového směrování byla nutná implementace multicastové směrovací tabulky a úprava existující implementace síťové vrstvy.

6.1.1 Implementace multicastové směrovací tabulky

K multicastové směrovací tabulce budou přistupovat různé moduly směrovače. Z toho důvodu jsme pro ni vytvořili samostatný modul `multicastRoutingTable`, který byl umístěn podobně jako unicastová směrovací tabulka přímo do modelu směrovače `ansaDualStackRouter` (obrázek 6.1). Nejedná se o modul komunikující pomocí zpráv, proto není propojen s žádným dalším modulem.

Implementace tabulky je rozdělena do tří tříd a koreluje implementace jiných již existujících tabulek podobného charakteru. Jeden záznam tabulky, představující multicastovou cestu, implementuje třída `MulticastIPRoute`. Parametry třídy byly vytvořeny podle návrhu (4.2.3) tak, aby se struktura tabulky shodovala s implementací od firmy Cisco.

```
class INET_API MulticastIPRoute : public cPolymorphic
{
   private:
       IPAddress source;            /**< Source of multicast */
       IPAddress group;             /**< Multicast group */
       IPAddress RP;                /**< Randevous point */
       std::vector<flag> flags;     /**< Route flags */
       // timers
       PIMgrt *grt;                 /**< Pointer to GraftRetryTimer*/
```



Obrázek 6.1: Model ansaDualStackRouter s označenými moduly, které byly přidány či modifikovány.

```
        PIMsat *sat;                /**< Pointer to SourceActiveTimer*/
        PIMsrt *srt;                /**< Pointer to StateRefreshTimer*/
        // interfaces
        inInterface inInt;          /**< Incoming interface */
        InterfaceVector outInt;     /**< Outgoing interface */
    ....
 };
```

Pro informace o vstupním rozhraní jsme pro větší přehlednost kódu vytvořili strukturu `inInterface` a pro odchozí rozhraní strukturu `outInterface`.

```
struct inInterface
{
   InterfaceEntry *intPtr;          /**< Pointer to interface */
   int intId;                       /**< Interface ID */
   IPAddress nextHop;               /**< RF neighbor */
};

struct outInterface
{
   InterfaceEntry *intPtr;          /**< Pointer to interface */
```

```
    int intId;                          /**< Interface ID */
    intState forwarding;                /**< Forward or Pruned */
    intState mode;                      /**< Dense, Sparse, ... */
    PIMpt *pruneTimer;                  /**< Pointer to PIM Prune Timer*/
    AssertState assert;                 /**< Assert state. */
};
typedef std::vector<outInterface> InterfaceVector;
```

Kromě uvedených parametrů obsahuje třída metody, které umožňují modifikaci, čtení a zápis hodnot parametrů. Vlastní tabulka je implementována třídou `MulticastRoutingTable`. Nejdůležitějším parametrem této třídy je vektor cest představující tabulku.

```
typedef std::vector<MulticastIPRoute *> RouteVector;

class INET_API MulticastRoutingTable: public cSimpleModule
{
    protected:
        RouteVector multicastRoutes; /**< Multicast routing table. */
        ...

    public:
        std::vector<MulticastIPRoute*> getRouteFor(IPAddress group);
        MulticastIPRoute *getRouteFor(IPAddress group, IPAddress source);
        std::vector<MulticastIPRoute*> getRoutesForSource(IPAddress source);
        void addRoute(const MulticastIPRoute *entry);
        bool deleteRoute(const MulticastIPRoute *entry);
        ...
};
```

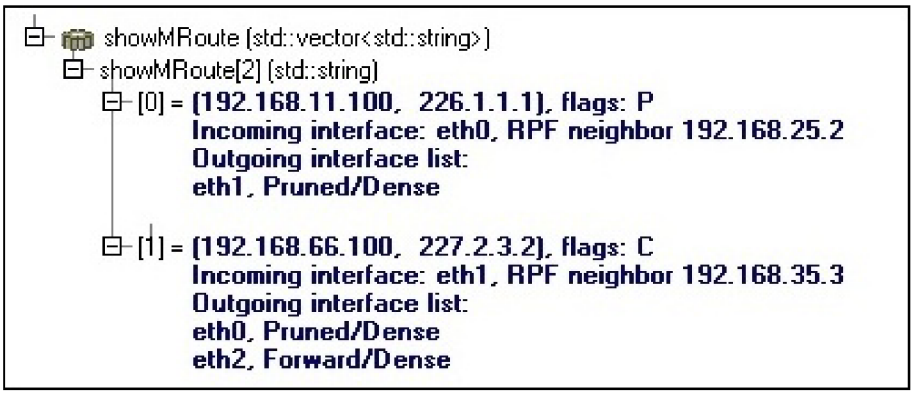
Pro vyhledávání cesty byly implementovány tři metody. Metoda `getRouteFor()` s jedním parametrem nalezne všechny cesty v tabulce pro zadanou multicastovou adresu. V případě, že metodě `getRouteFor()` předložíme adresu skupiny i zdroje, vrátí nám odkaz na jednu konkrétní cestu v tabulce. Pakliže budeme chtít zjistit všechny cesty pro zadaný zdroj multicástu, můžeme využít metodu `getRoutesForSource()`. Vkládání nových cest zajistí metoda `addRoute()` a odstranění metoda `deleteRoute()`.

Třída `MulticastRoutingTableAccess` umožňuje získání přístupu k tabulce. Jedinou metodu této třídy (`MulticastRoutingTableAccess()`) musí použít všechny třídy, které s multicastovou směrovací tabulkou chtějí pracovat. Metoda vyhledá instanci třídy v rámci modelu směrovače a vrátí na ní odkaz.

```
class INET_API MulticastRoutingTableAccess :
                                public ModuleAccess<MulticastRoutingTable>
{
    public:
        MulticastRoutingTableAccess() :
            ModuleAccess<MulticastRoutingTable>("multicastRoutingTable") {}
```

```
};
```

Textovou podobu multicastové tabulky z průběhu simulace můžete vidět na obrázku 6.2. Textový výstup byl přizpůsoben výstupu příkazu `show ip mroute` na směrovačích Cisco (obrázek 6.3). V naší tabulce se pouze nezobrazují časovače, protože textový výstup se aktualizuje pouze při změnách tabulky, a tak by mohly být hodnoty časovačů zavádějící.



Obrázek 6.2: Textová podoba multicastové tabulky.

```
(192.168.11.100, 226.1.1.1), 00:00:31/00:02:53, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet1/0, Forward/Dense, 00:00:31/00:00:00

(192.168.33.100, 227.2.3.2), 00:00:14/00:02:45, flags: PT
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list: Null
```

Obrázek 6.3: Multicastová tabulka z Cisco směrovače.

6.1.2 Modul síťové vrstvy podporující multicast

Existující implementace síťové vrstvy v podobě třídy `IP` se nechová k multicastovým datům korektně. Data rozesílá broadcastem, což neumožňuje korektní simulaci. Je také nutné, aby nová implementace síťové vrstvy spolupracovala s multicastovou směrovaní tabulkou.

Vzhledem k tomu, že implementační výstupy projektu ANSA nejsou zatím integrovány do knihovny INET, vytvořili jsme třídu `AnsaIP`, která dědí z třídy `IP`. V případě, že dojde ke změně ve třídě `IP`, změny se promítnou i do třídy `AnsaIP`.

```
class INET_API AnsaIP : public IP
{
   ...
   protected:
      void handlePacketFromNetwork(IPDatagram *datagram);
```

```
        void routeMulticastPacket(IPDatagram *datagram,
              InterfaceEntry *destIE, InterfaceEntry *fromIE);
    ...
};
```

Třída obsahuje dvě podstatné metody - `handlePacketFromNetwork()` a `routeMulticastPacket()`. Obě jsou již obsaženy ve třídě `IP`, ale bylo nutné je přepsat kvůli jejich zásadní roli při směrování multicastu.

V metodě `handlePacketFromNetwork()` přibyla podmínka, která automaticky odešle všechny pakety s protokolovým číslem 103 do modulu PIM. V metodě `routeMulticastPacket()` bylo nahrazeno vyhledávání v unicastové směrovací tabulce vyhledáváním v tabulce multicastové. V případě, že cesta v tabulce neexistuje, je přidána. Následně se data rozešlou na všechny rozhraní v seznamu odchozích rozhraní.

6.2 Implementace podpůrných prostředků pro protokol PIM

Ať už se rozhodneme využívat jakýkoliv režim protokolu PIM, neobejdeme se bez PIM zpráv (6.2.1), PIM časovačů (6.2.2), tabulky PIM rozhraní (6.2.3) a tabulky sousednosti (6.2.4).

6.2.1 PIM zprávy

Zprávy jsou definovány v souboru `PIMPacket.msg`. Jejich definice vychází z RFC, avšak byla odebrána některá nedefinovaná, nulová či další políčka, která nejsou pro simulaci důležitá. Přestože byl implementován pouze protokol PIM-DM, byly vytvořeny i definice zpráv, které využívá PIM-SM.

Po překladu se automaticky vygeneruje příslušný zdrojový a hlavičkový soubor obsahující třídu pro každý typ zprávy. Část `PIMPacket.msg` definující PIM hlavičku a PIM Hello zprávu vypadají následovně:

```
// Header
packet PIMPacket
{
   short       version = 2;
   short       type enum(PIMPacketType);
}

// Hello message
packet PIMHello extends PIMPacket
{
   short          type enum(PIMPacketType) = Hello;
   HelloEntry     helloContent[];
}
```

`PIMPacketType` je enumeration seznam obsahující hodnotu pro každý typ zprávy. `HelloEntry` je struktura pro hodnotu v Hello zprávě. Podle standartu by každá hodnota měla být v TLV formátu. Při simulování ale nemá žádný smysl udávat délku, proto struktura obsahuje jen typ a hodnotu.

6.2.2 PIM časovače

Jak jsme si uvedli v podsekci 2.2.1, protokol PIM-DM využívá poměrně dost časovačů. V diskrétní simulaci se časovače modelují jako zprávy, které objekt zasílá sám sobě se zadaným zpožděním. Jejich definici naleznete v souboru `PIMTimer.msg`. Struktura zprávy pro časovače Prune a Graft je následující:

```
// represents PIM Prune Timer
packet PIMpt extends PIMTimer
{
   timerKind = PruneTimer;          // Type of timer
   IPAddress      source;           // Source of multicast
   IPAddress      group;            // Multicast group address
   int            intId;            // ID of interface
}

// represents PIM Graft Retry Timer
packet PIMgrt extends PIMTimer
{
   timerKind = GraftRetryTimer;     // Type of timer
   IPAddress      source;           // Source of multicast
   IPAddress      group;            // Multicast group address
}
```

Každá zpráva obsahuje informaci o typu časovače. Tělo se odvíjí od toho, zda je časovač přiřazen cestě, která je definovaná IP adresou zdroje a multicastové skupiny, nebo rozhraní, u kterého je nutné navíc udávat jeho identifikátor.

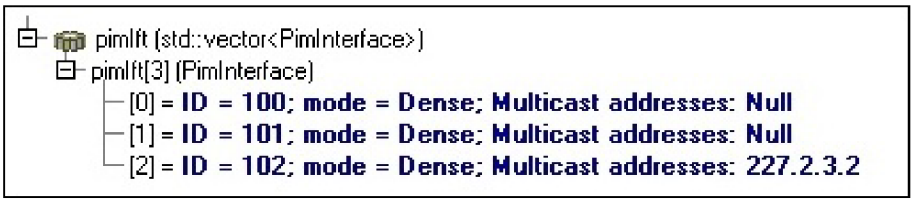
6.2.3 Modul tabulky PIM rozhraní

Tabulka rozhraní obsahuje statické informace o konfiguraci rozhraní načtené na začátku simulace z konfiguračního souboru. Podstatné informace pro každé rozhraní (viz 4.2.1) jsou uloženy v podobě parametrů třídy `PimInterface`. Tato třída implementuje jeden záznam tabulky.

```
class INET_API PimInterface: public cPolymorphic
{
   protected:
      int intID;              /**< Identification of interface. */
      InterfaceEntry *intPtr; /**< Pointer to interface table entry. */
      PIMmode mode;           /**< Type of mode. */
      /**< Multicast addresses assigned to interface. */
      std::vector<IPAddress>  intMulticastAddresses;
   ...
};
```

Samotná tabulka je implementována třídou `PimInterfaceTable`, jejíž nejdůležitější parametr je vektor ukazatelů na objekty `PimInterface`. Třída obsahuje základní metody pro přístup k jednotlivým záznamům tabulky a jejich přidání.

Podobně jak je tomu u multicastové směrovací tabulky, pro přístup k tabulce PIM rozhraní se používá třída `PimInterfaceTableAccess`. Textovou podobu tabulky si můžete prohlédnout na obrázku 6.4.



Obrázek 6.4: Textová podoba tabulky PIM rozhraní.

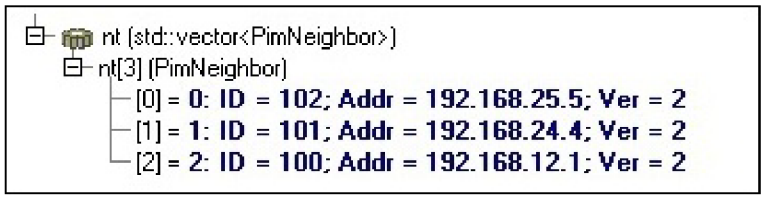
Na rozdíl od tabulky sousednosti, se kterou se běžně můžeme setkat na Cisco zařízeních, bylo do tabulky navíc přidáno pole multicastových IP adres, které jsou navázány na rozhraní. Při IGMP změně nám to pak umožní zjistit, jaké adresy byly na rozhraní přidány či z rozhraní odebrány (viz 6.3.1).

6.2.4 Modul tabulky sousednosti

Modul tabulky sousednosti byl naimplementován dle návrhu 4.2.2. Jednotlivé záznamy tabulky implementuje třída `PimNeighbor`.

```
class INET_API PimNeighbor: public cPolymorphic
{
   protected:
      int id;                    /**< Unique identifier of entry. */
      int intID;                 /**< Identification of interface. */
      InterfaceEntry *intPtr;    /**< Link to interface table entry. */
      IPAddress addr;            /**< IP address of neighbor. */
      int ver;                   /**< PIM version. */
      PIMnlt *nlt;               /**< Pointer to NeighborLivnessTimer.*/
   ...
};
```

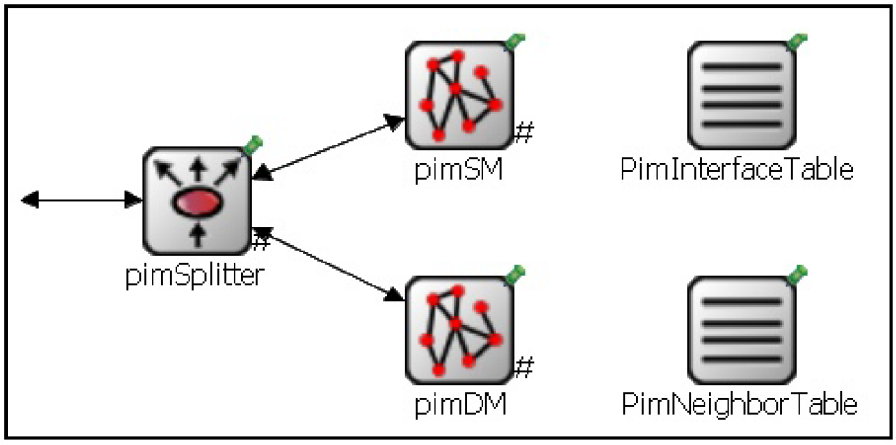
Podobně jako u tabulky PIM rozhraní (6.2.3) je tabulka sousednosti implementována třídou `PimNeighborTable` a pro přístup k ní se používá třída `PimNeighborTableAccess`. Textovou podobu tabulky sousednosti si můžete prohlédnout na obrázku 6.5.



Obrázek 6.5: Textová podoba tabulky sousednosti.

6.3 Implementace protokolu PIM

Protokol PIM je implementován dle návrhu 4.1 jako složený modul `pim` (obrázek 6.6) obsahující moduly pro PIM módy `pimDM` a `pimSM`, modul `PimSplitter`, moduly pro tabulku rozhraní `PimInterfaceTable` a tabulku sousednosti `PimNeighborTable`. Modul `pim` je napojen na modul síťové vrstvy `AnsaNetworkLayer`, jak můžete vidět na obrázku 6.1.



Obrázek 6.6: Složený model protokolu PIM se všemi svými komponentami.

6.3.1 Modul PIM Splitter

Jednoduchý modul `PimSplitter` je přímo připojen na modul síťové vrstvy `AnsaNetworkLayer`, od kterého dostává PIM zprávy (6.2.1). Ty pak buď zpracuje nebo přepošle do přímo připojených modulů `pimDM` nebo `pimSM`. Chování modulu implementuje třída `PimSplitter`. Její hlavičkový soubor je uveden v příloze C.1.

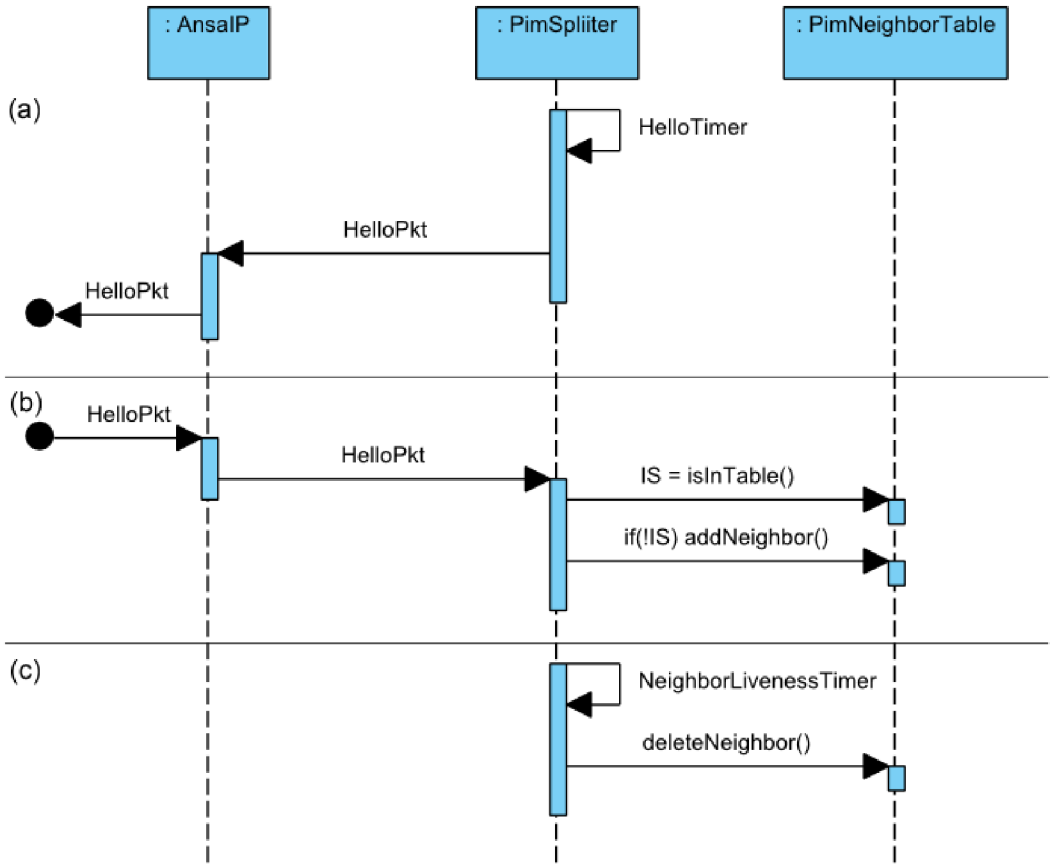
Udržování sousedství

Zpracování Hello zpráv a udržování sousedství je činnost shodná pro všechny režimy protokolu PIM. Z tohoto důvodu je Hello zpráva jediným typem zprávy, kterou Splitter nezasílá do konkrétního PIM modulu, ale rovnou ji zpracuje. Splitter je také jediným správcem tabulky sousednosti, ostatní moduly z ní pouze čtou.

Při inicializaci se nastaví časovač Hello na implicitní hodnotu 30 sekund. Každá zpráva je přijata metodou `handleMessage()`. Zde se zprávy třídí na vlastní (časovače) a cizí (PIM zprávy od jiných směrovačů). Pokud je časovač typu Hello, zavolá se metoda `sendHelloPkt()`, která pro každé rozhraní z tabulky rozhraní zavolá metodu `createHelloPkt()`. Ta vytvoří a odešle zprávu Hello (obrázek 6.7a).

V případě, že příchozí zpráva je PIM paket od jiného směrovače a je typu Hello, zavolá se pro její zpracování metoda `processHelloPkt()`. Pokud se jedná o jiný typ paketu, metoda `processPIMPkt()` jej odešle příslušnému PIM modulu.

Metoda `processHelloPkt()` zkontroluje, zda už se odesílatel nachází v tabulce sousednosti. Pokud ne, vytvoří pro něj nový záznam a vloží jej do tabulky (obrázek 6.7b). K záznamu také nastaví časovač Neighbor Liveness Timer. Pokud je soused v tabulce již zaznamenán, pouze resetuje Neighbor Liveness časovač.



Obrázek 6.7: Sekvenční diagram znázorňující (a) odesílání Hello zpráv sousedům, (b) zpracování přijatých Hello zpráv, (c) vypršení časovače Neighbor Liveness.

Po vypršení časovače Neighbor Liveness (trojnásobek časovače Hello) metoda `processNLTimer()` odstraní příslušný záznam z tabulky sousednosti (obrázek 6.7a).

Asynchronní události

Kromě synchronních událostí, kterými jsou příjmy zpráv, je nutná u protokolu PIM i obsluha těchto asynchronních událostí:

- nový multicastový datový tok,
- nový zájemce o multicastová data,
- data přijata na RPF rozhraní,
- data přijata na non-RPF rozhraní,
- data přijata na odříznutém (pruned) rozhraní,
- změna RPF rozhraní.

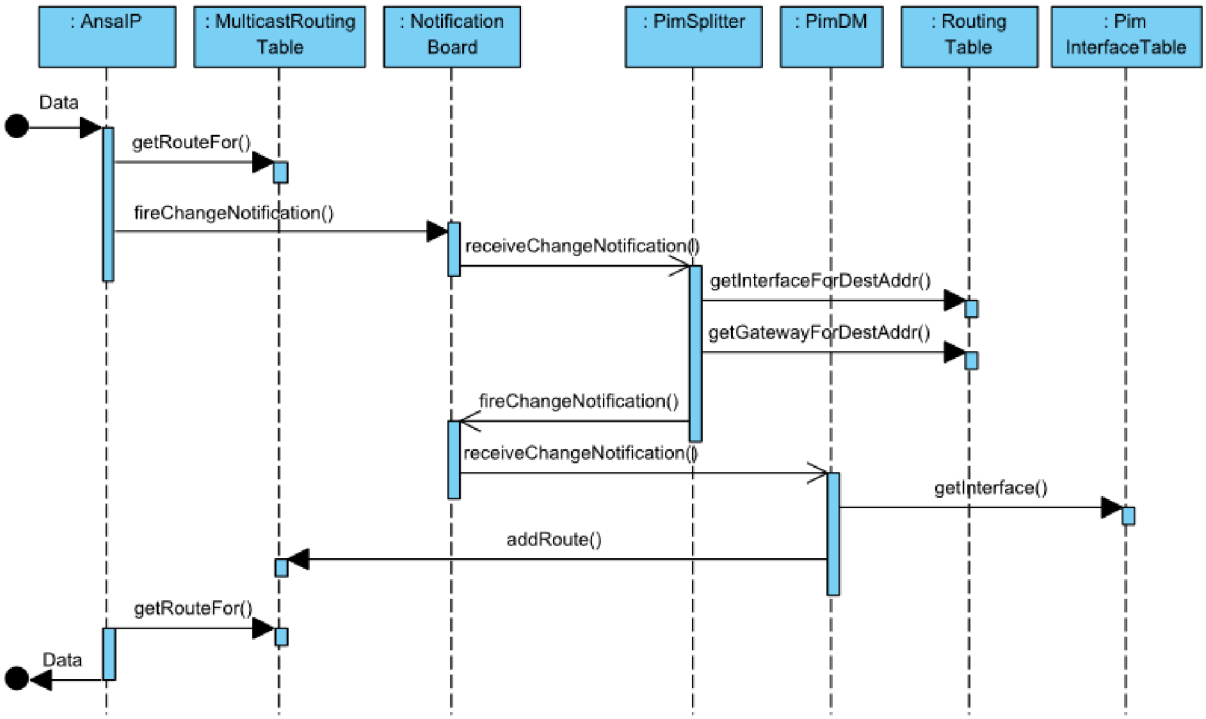
Pro upozornění na asynchronní událost využíváme s výhodou notifikační tabuli třídy `NotificationBoard` z knihovny INET. Ta pracuje se dvěma základními metodami. Metoda `fireChangeNotification()` na pomyslnou tabuli zapíše informaci o nějaké události. Pro příjem těchto informací musí zainteresovaná třída reimplementovat metodu `receiveChangeNotification()`.

Obě uvedené třídy mají dva parametry. První z nich je číslo, které identifikuje, o jakou událost se jedná, a druhý je odkaz na libovolný objekt, který nese přidanou informaci. Aby metoda `receiveChangeNotification()` věděla, na které události má reagovat, musí se třída přihlásit k odběru metodou `subscribe()` s parametrem v podobě identifikátoru vybrané události.

Zpracování asynchronních událostí

Modul PIM Splitter předzpracovává první dvě asynchronní události uvedené výše v seznamu. Nový multicastový datový tok zpracovává metoda `newMulticast()`. Metoda podle zdrojové IP adresy vyhledá v unicastové směrovací tabulce RPF rozhraní a zjistí IP adresu RPF souseda.

Na základě těchto informací vytvoří novou cestu (`MulticastIPRoute`) a vyvoláním nové asynchronní události ji předá k dalšímu zpracování konkrétnímu PIM modulu (obrázek 6.8). Celý proces vytváření nového záznamu do multicastové směrovací tabulky je zřejmý z diagramu 6.8.



Obrázek 6.8: Sekvenční diagram znázorňující vytvoření záznamu do multicastové směrovací tabulky po přijetí dat z nového multicastového toku.

Událost informující o změně multicastových adres přiřazených na rozhraní je předzpracovávaná metodou `igmpChange()`. Úkolem této metody je pouze zjistit, jaké multicastové

adresy byly na rozhraní přidány, případně byly z rozhraní odebrány. Seznam takových IP adres přepošle k dalšímu zpracování konkrétnímu PIM modulu.

6.3.2 Modul pimDM

Jednoduchý modul `pimDM` je připojen k modulu `PimSplitter` (viz 6.3.1), přes který může zprostředkovaně komunikovat se síťovou vrstvou. Chování modulu implementuje třída `pimDM`. Její hlavičkový soubor je uveden v příloze C.2.

Vzhledem k podstatě simulátoru, který reaguje na příchozí zprávy, nebylo možné implementovat konečné automaty protokolu PIM-DM zcela průhledně. Jednotlivé kroky automatů mezi stavy jsou rozesety do různých metod třídy. Stejně tak jedna metoda provádí přechody ve více konečných automatech.

Jedním z cílů této práce bylo přiblížení naší implementace implementaci protokolu PIM na Cisco směrovačích. Z tohoto důvodu používáme jiné označení stavů než RFC 3973 a některé stavy se nikde explicitně nezapisují, ale jsou zřejmé z okolností (běžící časovač, (ne)nastavení příznaku).

Zpracování asynchronních události

Modul pimDM musí zpracovat všechny asynchronní události uvedené v části 6.3.1. Událost o novém multicastovém datovém toku zpracovává metoda `newMulticast()`. PIM Splitter modulu pimDM předá již částečně vyplněný záznam. Metoda doplní seznam odchozích rozhraní a přidá cestu metodou `addRoute()` třídy `MulticastRoutingTable` do multicastové směrovací tabulky (viz diagram 6.8).

K cestě také nastaví časovač Source Active Timer (`createSourceActiveTimer()`). Zde byla provedena změna oproti konečnému automatu 2.2.6. Source Active Timer měl být nastaven pouze u směrovače přímo připojeného ke zdroji multicastu. Po jeho vypršení metoda `processSourceActiveTimer()` vymaže cestu z tabulky. Tuto činnost je ale nezbytně nutné provádět u všech směrovačů, ne jen u přímo připojeného.

Směrovač přímo připojený ke zdroji si k cestě navíc nastaví State Refresh Timer (`createStateRefreshTimer()`). Jakmile modul příjme zprávu o vypršení časovače, zavolá metodu `processStateRefreshTimer()`, která zajistí vyslání zprávy State Refresh (`sendPimStateRefresh()`) na všechna odchozí rozhraní. V každé zprávě se nastaví Prune Indicator buď na 1, pokud je rozhraní Pruned, nebo 0, pokud je ve stavu Forward.

Každá cesta má příznak (flag), který se používá k označení jejího stavu. Po vzoru Cisco používáme P pro odříznutou cestu a C pro přímo připojené odběratele multicastu. Navíc jsme si přidali vlastní příznak A, který označuje, že zdroj multicastu je přímo připojený k směrovači (ekvivalent stavu Originator, 2.2.6).

Přijetí dat na RPF rozhraní vyvolá událost, která je zpracována metodou `dataOnRpf()` Ta pouze restartuje časovač Source Active.

Pokud na non-RPF rozhraní point-to-point linky přijdou data, zavolá se metoda `dataOnNonRpf()`, která na rozhraní odešle JoinPrune zprávu. Stav rozhraní je poté změněn na Pruned. Pokud neběží časovač Graft Retry, metoda `dataOnPruned()` reaguje na přijetí dat na rozhraní ve stavu Pruned také odesláním JoinPrune zprávy.

Speciálním případem je událost vyvolaná změnou RPF rozhraní, kterou zpracovává metoda `rpfIntChange()`. Metoda nalezne ve směrovací tabulce nové RPF rozhraní a odstraní ho ze seznamu odchozích rozhraní. Tam je naopak třeba přidat původní RPF rozhraní, které nastaví svůj stav na Forward. Pokud směrovač nebyl odříznut od stromu, pokusí se znovu připojit vysláním Graft zprávy na nové RPF rozhraní.

Metoda `newMulticastAddr()` se zabývá přidáním nových multicastových IP adres na rozhraní. Jejich seznam dostane už předem připravený od modulu Splitter. V multicastové směrovací tabulce vyhledá všechny cesty s danou multicastovou adresou a přidá jim nové rozhraní do seznamu odchozích rozhraní. Opačně funguje metoda `oldMulticastAddr()`, která prochází seznam adres, které byly z rozhraní odstraněny.

Vyprázdnění a naplnění seznamu odchozích rozhraní

Po té, co je nová cesta přidána do multicastové směrovací tabulky, nachází se v pomyslném stavu Forwarding, tzn. cesta nemá příznak P (Pruned). Jakmile nastane situace, že se seznam odchozích rozhraní vyprázdní (nebo jsou všechna rozhraní ve stavu Pruned), zavolá se metoda `sendPimJoinPrune()`, která vytvoří JoinPrune zprávu a odešle ji na RPF rozhraní. Cestě je přidán příznak P a automat tak přejde do stavu Pruned.

K vyprázdnění seznamu odchozích rozhraní může dojít v těchto situacích:

- Metoda `processPrunePacket()` zpracovávající žádost downstream směrovače o odříznutí od stromu změní stav rozhraní na Pruned.
- Metoda `oldMulticastAddr()` zpracovávající událost odhlášení přímo připojeného odběratele multicastu odebere rozhraní ze seznamu.
- Metoda `newMulticast()`, která se stará o přidání nové cesty do multicastové směrovací tabulky, zjistí, že na žádném non-RPF rozhraní není připojen PIM soused.

Původně prázdný seznam odchozích rozhraní se později může znovu naplnit. Modul pak odešle na své RPF rozhraní zprávu Graft (`sendPimGraft()`) a nastaví časovač Graft Retry Timer (`createGraftRetryTimer()`). Pokud časovač vyprší, zavolá se metoda `processGraftRetryTimer()`, která provede znovu odeslání zprávy Graft.

Ke znovu naplnění seznamu odchozích rozhraní může dojít v těchto situacích:

- Metoda `processGraftPacket()` zpracovávající žádost downstream směrovače o připojení k stromu změní stav rozhraní na Forward.
- Metoda `newMulticastAddr()` zpracovávající událost přihlášení přímo připojeného odběratele multicastu přidá rozhraní na seznam.
- Metoda `processPruneTimer()`, která se stará o expiraci časovače Prune, změní stav rozhraní na Forward.

Zpracování externích zpráv

Zprávy od jiných PIM směrovačů jsou nejprve přijaty modulem PimSplitter a následně odeslány do modulu pimDM. Metoda `handleMessage()` rozliší, jestli se jedná o vlastní zprávu (časovač) nebo o zprávu externí. Většina zpráv se dále zasílá metodě `processJoinPruneGraftPacket()`. Ta parsuje zprávy PruneJoin, Graft i Graft Ack, protože mají stejnou vnitřní strukturu.

Uvažujme, že zpráva JoinPrune přijatá na non-PRF rozhraní obsahuje zdrojovou a multicastovou adresu definující strom, od kterého se chce downstream směrovač odříznout. Tento požadavek zpracuje metoda `processPrunePacket()`. Rozhraní, na které přišla taková zpráva, změní svůj stav na Pruned. Ke každému odchozímu rozhraní, které je v Pruned stavu, se nastaví časovač Prune metodou `createPruneTimer()`.

Pokud rozhraní bylo ve stavu Pruned ještě před přijetím zprávy, pouze se časovač restartuje. Vypršení časovače zpracovává metoda `processPruneTimer()`. Ta nastaví rozhraní zpátky do stavu Forwarding.

Příchozí zpráva Graft je nejprve parsována metodou `processJoinPruneGraftPacket()` a následně zpracována metodou `processGraftPacket()`. Ta odpoví downstream směrovači zprávou Graft Ack (`sendPimGraftAck()`). Je-li rozhraní ve stavu Pruned, změní se jeho stav na Forwarding a zastaví se časovač Prune Timer.

Při přijetí zprávy Graft Ack na RPF rozhraní se zavolá metoda `processGraftAckPacket()`. Ta zastaví časovač Graft Retry a smaže příznak P ze záznamu cesty. Stejné akce provede metoda `processStateRefreshPacket()` po přijetí zprávy State Refresh, pokud je Prune Indicator zprávy nastaven na 0.

Metoda `processStateRefreshPacket()` následně začne procházet seznam odchozích rozhraní. Každému rozhraní, které je ve stavu Pruned, se restartuje časovač Prune a odešle se na něj State Refresh zpráva metodou `sendPimStateRefresh()`. Ve zprávě se nastaví Prune Indicator buď na 1, pokud je rozhraní Pruned, nebo 0, pokud je ve stavu Forward.

Problematika šíření multicastu na L2 vrstvě

Situaci, kdy je k jedné LAN síti připojeno více směrovačů, řeší Assert Message State Machine (viz 2.2.7). Aby mohl být implementován, je třeba, aby simulátor splňoval několik požadavků.

Ten nejzásadnější je schopnost směrovače rozeznat, zda je na rozhraní point-to-point linka nebo LAN. Bohužel v knihovně INET ani ANSAINET není žádný prostředek, kterým by bylo možné toto určit. Existuje možnost zapsat typ linky do konfigurace. Avšak to není ideální řešení, protože protokol PIM na Cisco směrovačích nemá žádný takový konfigurační příkaz. Vzhledem k tomu, že se do budoucna předpokládá vytváření konfiguračních souboru pro simulátor přímo z reálných Cisco konfigurací, vnášelo by přidání vlastních příkazů zbytečné problémy.

Nejjednodušší, avšak nikoli stoprocentní způsob, jak zjistit typ připojené linky, je podle množství PIM sousedů na jednom rozhraní. Do třídy `PimNeighborTable` byla proto přidána metoda `getNumNeighborsOnInt()`, která pro zadaný identifikátor rozhraní vrátí k němu připojený počet PIM sousedů.

V době, kdy vzniklo RFC, se zřejmě uvažovalo o připojení LAN sítě sběrnicovou topologií realizovanou koaxiálním kabelem. LAN sítí je tedy myšlena jedna broadcastová doména. Dnes je možné připojení LAN pomocí přepínače či rozbočovače. Zde tkví také druhý zásadní problém.

Multicast, tak jak byl implementován v rámci této práce, se zaměřil na L3 vrstvu. Pro správné fungování multicastu na L2 vrstvě je nutné tuto podporu doimplementovat. Chybí zde totiž především užívání multicastových MAC adres a také implementace multicastu do modelu přepínače.

Kdybychom tedy provedli poměrně náročnou implementaci Assert Message konečného automatu a s tím související další implementační kroky protínající i downstream a upstream automaty, multicast by se stejně šířil LAN sítí pomocí broadcastu. A to není rozhodně očekávaným cílem našich simulací.

Z těchto důvodů není automat Assert Message State Machine zcela implementován. Implementován byl speciální případ vysílání na sdíleném médiu, kterým je přijetí multicastových dat na non-RPF rozhraní připojené k point-to-point lince. K tomu může dojít, pokud se mezi směrovači vytvoří smyčka. Metoda `dataOnNonRpf()` zareaguje na událost

odesláním zprávy PruneJoin na takové rozhraní a jeho stav změní na Pruned.

V kódu jsou označena místa, která bude nutné doplnit pro plnou funkcionalitu automatu, až bude funkční i L2 multicast. Implementace podpory pro L2 multicast je ale nad rámec této práce. Směrování multicastového provozu jako takové je funkční a bylo odzkoušeno simulacemi na několika topologiích.

6.4 Shrnutí

Přestože úkolem této práce byla implementace multicastového směrování, bylo nutné začít s implementací základní podpory multicastu jako takového. Představili jsme si proto implementaci směrovací tabulky a úpravu třídy implementující síťovou vrstvu, která nyní již umí směrovat multicastový datový provoz.

Následně jsme se zaměřili na implementaci protokolu PIM. Tento protokol potřebuje kromě implementace vlastního směrovacího protokolu také podpůrné prostředky. Ukázali jsme si způsob, jakým byla implementována tabulka PIM rozhraní, tabulka sousednosti, PIM zprávy a PIM časovače.

Podle návrhu jsme implementovali prvek PIM Splitter, který umožňuje snadnější rozšíření o podporu dalších PIM protokolů. Kromě způsobu, jakým pracuje s příchozími zprávami, jsme si ukázali, jak zpracovává asynchronní události. Nejnáročnější byla implementace režimu dense mode. V kapitole je popsáno, jak byly implementovány jednotlivé části návrhu tohoto protokolu i nesnáze, které přinesla slabá podpora multicastu na L2 vrstvě.

Implementace všech zmíněných částí byla poměrně náročná a výsledkem je velké množství tříd a zdrojových souborů. Pro lepší přehlednost byla vygenerována programová dokumentace (viz příloha B) a diagram tříd (příloha C.3). Implementace byla odzkoušena na několika příkladech, které jsou nyní součástí knihovny ANSAINET.

Kapitola 7

Porovnání simulace s chováním reálné sítě

V závěrečné kapitole provedeme porovnání chování simulace a reálné sítě. Zaměříme se na PIM zprávy, které si mezi sebou vyměňují směrovače, na obsahy tabulek rozhraní, sousednosti a multicastových směrovacích tabulek.

7.1 Návrh scénáře

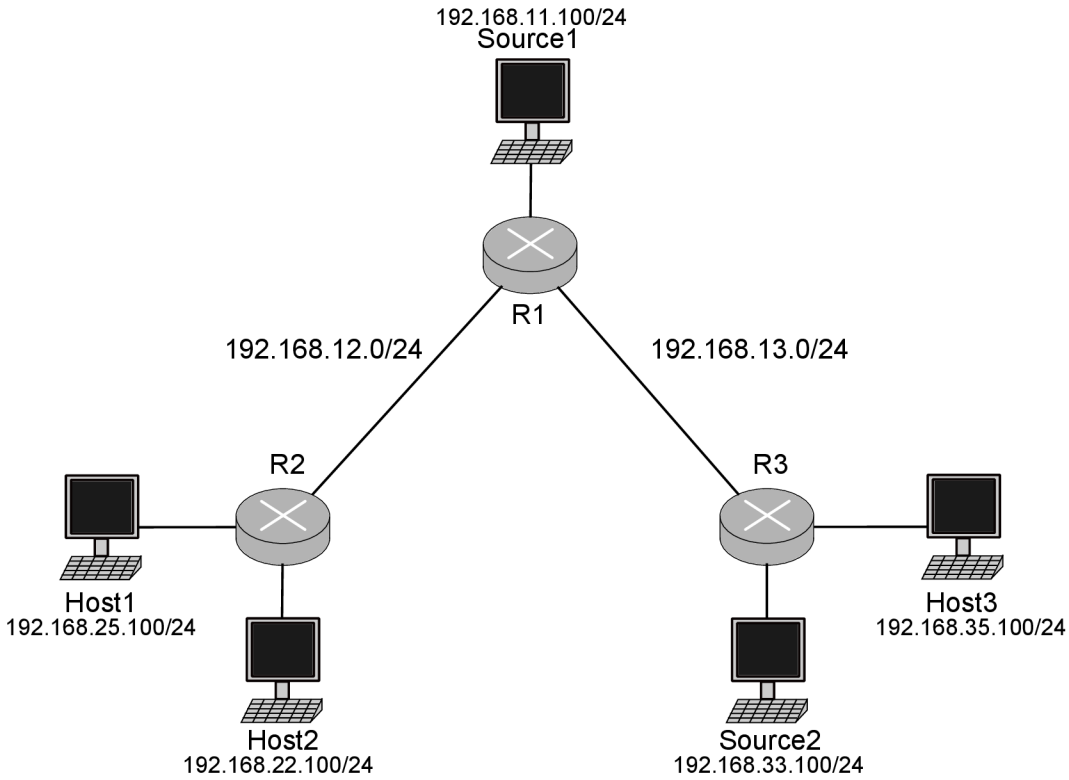
Přestože byla simulace testována i na rozsáhlejších topologiích, rozhodli jsme se pro porovnání zvolit trochu jednodušší síť. Je to zejména kvůli náročnosti odchytávání paketů v reálné síti a velikosti výstupů. Síť se skládá ze tří směrovačů propojených do stromu.

Dále se v síti nacházejí tři hosté, kteří mají různé požadavky na přijímání multicastu, a dva zdroje multicastových dat. Propojení síťových prvků a adresaci můžete vidět na obrázku 7.1

Tabulka 7.1 popisuje scénář přihlašování (S) a odhlašování (E) hostů spolu s přibližnými časy, kdy zdroje začínají (S) a končí (E) vysílání. Reálná síť je vybudována z Cisco směrovačů C3640 s IOS 12.4(16). Pro odchytávání paketů je použit program Wireshark.

Č.	**Čas (s)**	**Síťový prvek**	**Multicastová skupina**	**Akce**
1	0	Host1	226.2.2.2	S
2	87	Source1	226.1.1.1	S
3	144	Host2	226.1.1.1	S
4	215	Source2	226.2.2.2	S
5	264	Host3	226.1.1.1	S
6	323	Host2	226.2.2.2	S
7	364	Host2	226.1.1.1	E
8	399	Source2	226.2.2.2	E
9	619	Source2	226.1.1.1	S

Tabulka 7.1: Scénář přihlašování/odhlašování příjemců a vysílání zdrojů.



Obrázek 7.1: Návrh sítě používaná při testování.

7.2 Porovnání získaných dat

Porovnávat budeme jak pakety, které si směrovače mezi sebou posílají, tak screenshoty směrovacích tabulek. Z Cisco tabulek byly vypuštěny záznamy pro sparse mode, které jsou pro naše účely zbytečné. Abychom práci nezahltili velkým počtem obrázků, jsou zde uváděny jen příklady ze směrovačů, na kterých došlo k nějaké zajímavé změně.

Ve Wiresharku jsme vyfiltrovali PIM zprávy a exportovali jsme je do textové podoby pro jednodušší prezentaci. V prvním sloupečku výstupu je číslo paketu, následuje čas zachycení počítaný od spuštění záznamu, IP adresa odesílatele, IP adresa příjemce, protokol (PIMv2) a typ zprávy. Vzhledem k tomu, že záznam byl na linkách spuštěn s několika sekundovým zpoždění, časy uvedené u zpráv se mírně liší.

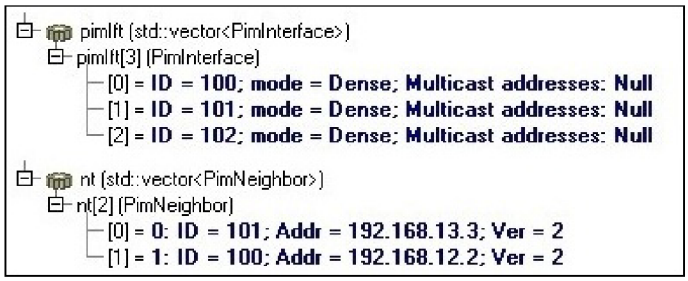
U simulace se na konzoli vypisovaly informace o zprávách, které daný směrovač přijal (odeslané se kvůli duplikaci nevypisovali). Ve výpisech se nejprve objevuje simulační čas následován názvem směrovače, v závorce je uvedeno, na kterém rozhraní zprávu přijal, za typem zprávy se v závorce nachází IP adresa odesílatele a IP adresa příjemce.

7.2.1 1. krok

V prvním kroku se Host1 přihlásí k odběru multicastových dat (226.2.2.2). Tento krok nevyvolal žádnou PIM zprávu ani změnu ve směrovací tabulce. Můžeme si alespoň porovnat výstupy tabulek rozhraní a tabulek směrovačů (obrázky 7.2 a 7.3).

Sítí se zatím šíří jen Hello pakety. Zprávy přijaté směrovači při simulaci:

```
30.101098715601 R1(101): PIMHello (192.168.13.3, 224.0.0.13)
```



Obrázek 7.2: Tabulka sousedství a rozhraní směrovače R1 ze simulace.

```
R1#sh ip pim int
Address          Interface                Ver/   Nbr    Query  DR     DR
                                          Mode   Count  Intvl  Prior
192.168.12.1     FastEthernet0/0          v2/D   1      30     1      192.168.12.2
192.168.13.1     FastEthernet0/1          v2/D   1      30     1      192.168.13.3
192.168.11.1     FastEthernet0/2          v2/D   0      30     1      192.168.11.1
R1#sh ip pim neigh
PIM Neighbor Table
Neighbor          Interface                Uptime/Expires    Ver   DR
Address                                                            Prio/Mode
192.168.12.2      FastEthernet0/0          00:04:02/00:01:38 v2    1 / DR S
192.168.13.3      FastEthernet0/1          00:04:02/00:01:39 v2    1 / DR S
```

Obrázek 7.3: Tabulka sousedství a rozhraní směrovače R1 z reálné sítě.

```
30.435653204837 R2(100): PIMHello (192.168.12.1, 224.0.0.13)
30.435653204837 R3(100): PIMHello (192.168.13.1, 224.0.0.13)
33.240866102784 R1(100): PIMHello (192.168.12.2, 224.0.0.13)
```

Zprávy zachycené na lince R1-R2 na reálné síti:

```
7 23.592000   192.168.12.2          224.0.0.13        PIMv2    Hello
8 25.461000   192.168.12.1          224.0.0.13        PIMv2    Hello
```

Zprávy zachycené na lince R1-R3 na reálné síti:

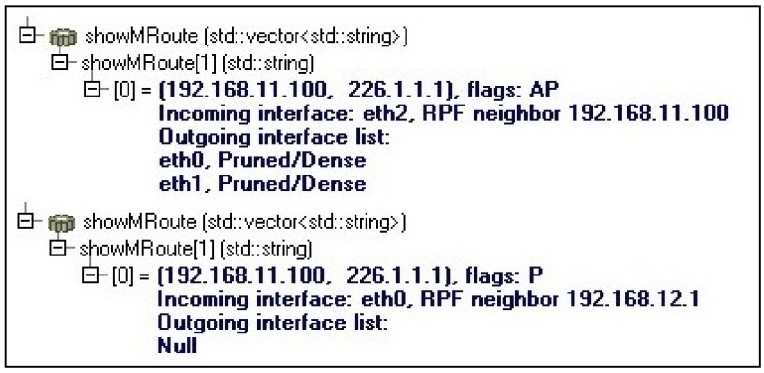
```
5 12.008000   192.168.13.1          224.0.0.13        PIMv2    Hello
6 13.102000   192.168.13.3          224.0.0.13        PIMv2    Hello
```

7.2.2 2. krok

Sítí se začala šířit multicastová data pro skupinu 226.1.1.1. Každý směrovač si pro skupinu vytvoří záznam do tabulky. Do skupiny ale není nikdo přihlášen, takže se všechny směrovače postupně odřežou (obrázky 7.4 a 7.5).

Zprávy přijaté směrovači při simulaci:

```
Send: appData-0 at time = 87
87.000031999997 R1(100): PIMJoinPrune (192.168.12.2, 224.0.0.13)
```



Obrázek 7.4: Multicastové směrovací tabulky směrovače R1 (nahoře) a R2 (dole) ze simulace (krok 2).

```
R1#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:00:23/00:02:36, flags: PT
  Incoming interface: FastEthernet2/0, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Prune/Dense, 00:00:22/00:02:41
    FastEthernet0/1, Prune/Dense, 00:00:22/00:02:40
R2#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:00:31/00:02:28, flags: PT
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list: Null
```

Obrázek 7.5: Multicastové směrovací tabulky směrovače R1 (nahoře) a R2 (dole) z reálné sítě (krok 2).

```
87.000031999997 R1(101): PIMJoinPrune (192.168.13.3, 224.0.0.13)
```

Zprávy zachycené na lince R1-R2 na reálné síti:

```
34 87.157000   192.168.12.2      224.0.0.13       PIMv2    Join/Prune
36 89.012000   192.168.12.2      224.0.0.13       PIMv2    Join/Prune
```

Zprávy zachycené na lince R1-R3 na reálné síti:

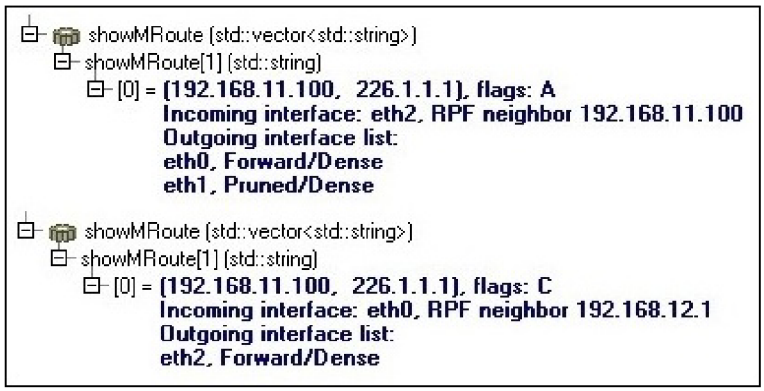
```
32 75.664000   192.168.13.3      224.0.0.13       PIMv2    Join/Prune
34 79.138000   192.168.13.3      224.0.0.13       PIMv2    Join/Prune
```

Směrovač R1 odeslal další data dříve, než přijal JoinPrune zprávu od směrovače R2(R3), a tak směrovač R2(R3) poslal další zprávu JoinPrune. Při simulaci byly pakety generovány jen jednou za 5 sekund, takže JoinPrune zpráva se na každé lince objevila pouze jedenkrát.

7.2.3 3. krok

Host2 se přihlásil do skupiny 226.1.1.1. Směrovač R2 pro ni má v tabulce záznam, je ale od stromu odříznutý. Znovu se ke stromu připojí vysláním zprávy Graft RPF sousedovi (R1).

Ke změně dojde ve směrovací tabulce směrovačů R1 a R2 (obrázky 7.6 a 7.7).



Obrázek 7.6: Multicastové směrovací tabulky směrovače R1 (nahoře) a R2 (dole) ze simulace (krok 3).

```
R1#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:02:54/00:02:50, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:00:47/00:00:00
    FastEthernet0/1, Prune/Dense, 00:02:54/00:00:06
R2#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:02:27/00:02:54, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:00:25/00:00:00
```

Obrázek 7.7: Multicastové směrovací tabulky směrovače R1 (nahoře) a R2 (dole) z reálné sítě (krok 3).

Zprávy přijaté směrovači při simulaci:

```
144.000026879996 R1(100): PIMGraft (192.168.12.2, 192.168.12.1)
144.000033599995 R2(100): PIMGraftAck (192.168.12.1, 192.168.12.2)
Send: appData-6 at time = 147
Arrive: appData-6 at time = 147.000025279998
```

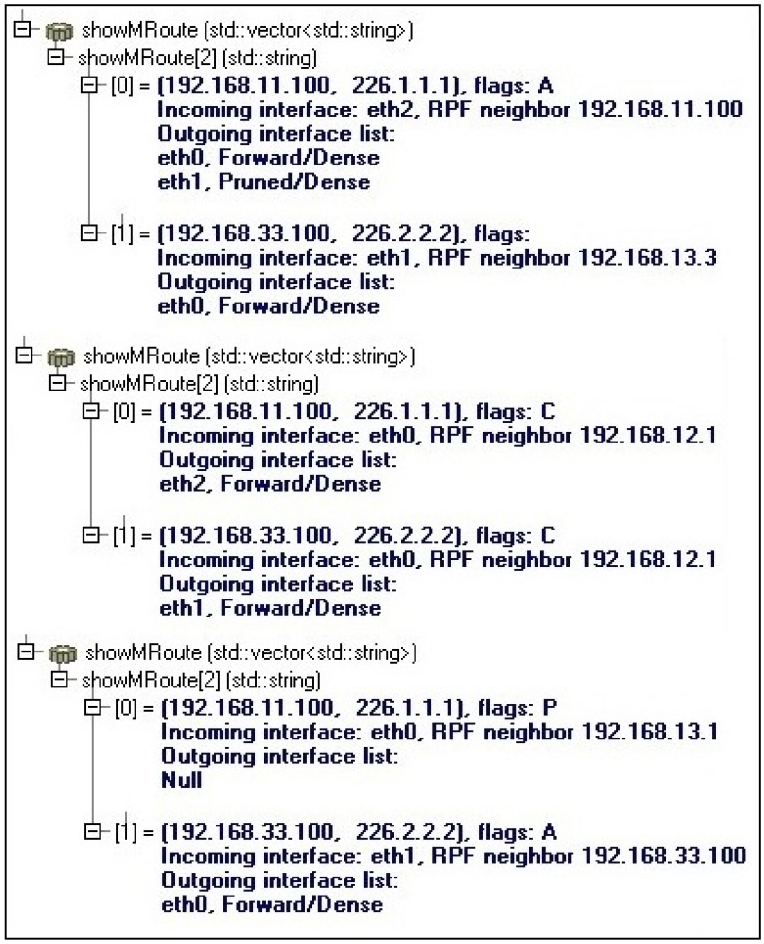
Zprávy zachycené na lince R1-R2 na reálné síti:

```
59 144.406000  192.168.12.2      192.168.12.1        PIMv2    Graft
60 144.440000  192.168.12.1      192.168.12.2        PIMv2    Graft-Ack
```

7.2.4 4. krok

Sítí se začne šířit nový multicastový datový proud s cílovou IP adresou 226.2.2.2. Všechny směrovače v síti si vytvoří nový záznam v směrovací tabulce (obrázky 7.8 a 7.9). V síti již

existuje příjemce (Host 1, viz 7.2.1), který začne data okamžitě odebírat. Nedojde k žádné výměně PIM zpráv.



Obrázek 7.8: Multicastové směrovací tabulky směrovačů (shora) R1, R2 a R3 ze simulace (krok 4).

7.2.5 5. krok

Host3 má zájem o data ze skupiny 226.1.1.1. Směrovač R3 byl do té doby odřezaný od multicastového stromu. Znovu se k němu připojí vysláním zprávy Graft. Směrovač R1 nyní odesílá data oběma svým sousedům (obrázky 7.10 a 7.11).

Zprávy přijaté směrovači při simulaci:

```
264.000026879996 R1(101): PIMGraft (192.168.13.3, 192.168.13.1)
264.000033599995 R3(100): PIMGraftAck (192.168.13.1, 192.168.13.3)
```

Zprávy zachycené na lince R1-R3 na reálné síti:

```
505 264.965000  192.168.13.3     192.168.13.1       PIMv2    Graft
506 265.005000  192.168.13.1     192.168.13.3       PIMv2    Graft-Ack
```

```
R1#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:08:44/00:01:20, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:06:37/00:00:00
    FastEthernet0/1, Prune/Dense, 00:02:32/00:00:36

(192.168.33.100, 226.2.2.2), 00:02:03/00:02:55, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.3
  Outgoing interface list:
    FastEthernet0/0, Forward/Dense, 00:02:03/00:00:00
```

```
R2#sh ip mroute

(192.168.11.100, 226.1.1.1), 00:02:27/00:02:54, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:00:25/00:00:00

(192.168.33.100, 226.2.2.2), 00:06:27/00:02:58, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/2, Forward/Dense, 00:06:28/00:00:00
```

```
R3#sh ip mroute

(192.168.11.100, 226.1.1.1), 00:01:33/00:01:26, flags: PT
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.1
  Outgoing interface list: Null

(192.168.33.100, 226.2.2.2), 00:00:59/00:02:55, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/1, Forward/Dense, 00:00:59/00:00:00
```

Obrázek 7.9: Multicastové směrovací tabulky směrovačů (zhora) R1, R2 a R3 z reálné sítě (krok 4).

7.2.6 6. krok

Host2 se stane členem obou skupin - 226.1.1.1 (viz 7.2.3) a nově 226.2.2.2. Směrovač R2 provede změnu ve směrovací tabulce (obrázky 7.12 a 7.13), ale zprávu odesílat žádnou nemusí, protože k multicastovému stromu je již připojen (viz 7.2.4).

7.2.7 7. krok

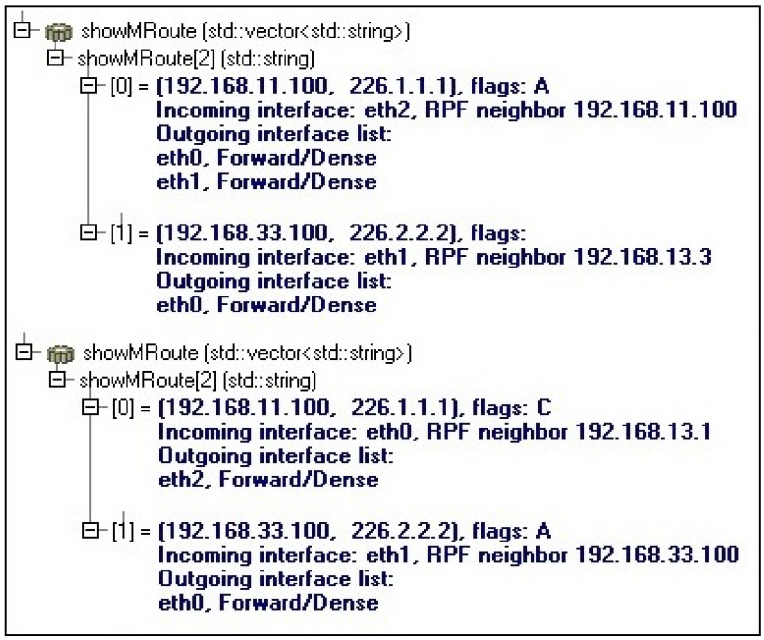
Host2 již nemá zájem o data z multicastové skupiny 226.1.1.1. Vzhledem k tomu, že byl jediným odběratelem této skupiny na směrovači R2, směrovač se odřízne od multicastového stromu odesláním zprávy JoinPrune směrovači R1. U něj to také vyvolá změnu ve směrovací tabulce (obrázky 7.14 a 7.15).

Zprávy přijaté směrovači při simulaci:

```
366.000013439998 R1(100): PIMJoinPrune (192.168.12.2, 224.0.0.13)
```

Zprávy zachycené na lince R1-R2 na reálné síti:

```
2601 364.496000  192.168.12.2       224.0.0.13     PIMv2    Join/Prune
```



Obrázek 7.10: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) ze simulace (krok 5).

```
(192.168.11.100, 226.1.1.1), 00:03:03/00:02:58, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:03:03/00:00:00

(192.168.33.100, 226.2.2.2), 00:06:27/00:02:58, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/2, Forward/Dense, 00:06:28/00:00:00
```

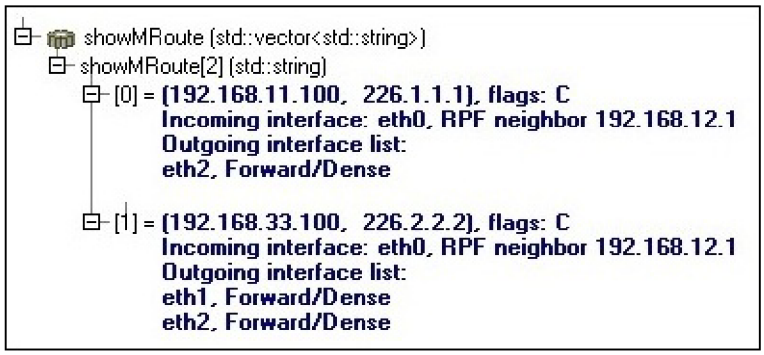
```
(192.168.11.100, 226.1.1.1), 00:03:13/00:02:58, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:00:36/00:00:00

(192.168.33.100, 226.2.2.2), 00:06:37/00:02:57, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/1, Forward/Dense, 00:06:38/00:00:00
```

Obrázek 7.11: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) z reálné sítě (krok 5).

7.2.8 8. krok

Zdroj Source2 přestane šířit data multicastové skupiny 226.2.2.2. Tato změna se ale v síti projeví až po 180s, po kterých si všechny směrovače vymažou záznam pro skupinu z tabulky (obrázky 7.16 a 7.17). V síti nedojde k žádné výměně PIM zpráv.

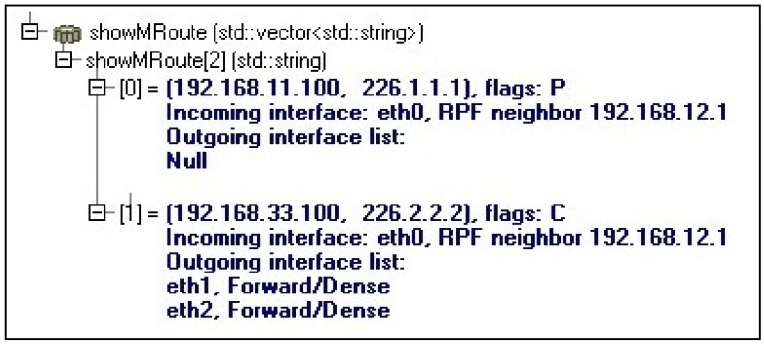


Obrázek 7.12: Multicastová směrovací tabulka směrovače R2 ze simulace (krok 6).

```
(192.168.11.100, 226.1.1.1), 00:10:31/00:02:31, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:10:31/00:00:00

(192.168.33.100, 226.2.2.2), 00:13:56/00:02:50, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:00:59/00:00:00
    FastEthernet0/2, Forward/Dense, 00:13:56/00:00:00
```

Obrázek 7.13: Multicastová směrovací tabulka směrovače R2 z reálné sítě (krok 6).

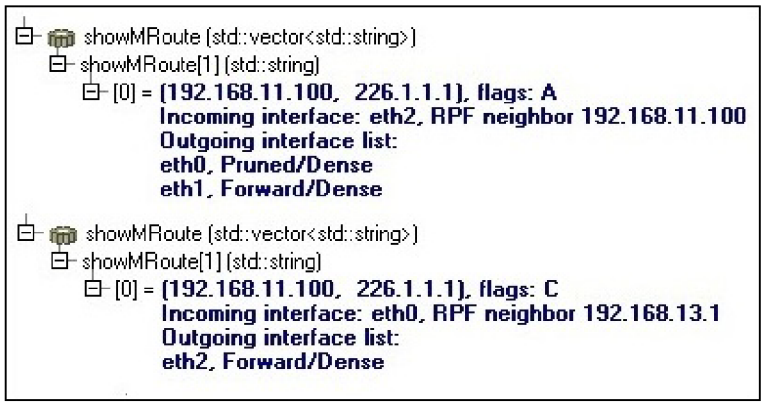


Obrázek 7.14: Multicastová směrovací tabulka směrovače R2 ze simulace (krok 7).

```
(192.168.11.100, 226.1.1.1), 00:02:29/00:02:29, flags: PT
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list: Null

(192.168.33.100, 226.2.2.2), 00:19:57/00:02:59, flags: T
  Incoming interface: FastEthernet0/0, RPF nbr 192.168.12.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:07:01/00:00:00
    FastEthernet0/2, Forward/Dense, 00:19:58/00:00:00
```

Obrázek 7.15: Multicastová směrovací tabulka směrovače R2 z reálné sítě (krok 7).



Obrázek 7.16: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) ze simulace (krok 8).

```
R1#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:12:47/00:02:54, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Prune/Dense, 00:01:27/00:01:41
    FastEthernet0/1, Forward/Dense, 00:12:47/00:00:00
```

```
R3#sh ip mroute
(192.168.11.100, 226.1.1.1), 00:12:08/00:02:58, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:12:08/00:00:00
```

Obrázek 7.17: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) z reálné sítě (krok 8).

7.2.9 9. krok

Po čase začne Source2 znovu vysílat, tentokrát do multicastové skupiny 226.1.1.1. Do této skupiny již vysílá Source1 (viz 7.2.2). Všechny směrovače si vytvoří v tabulce nový záznam s novým zdrojem (obrázky 7.18 a 7.19). O data z této skupiny má zájem pouze Host3 připojený k směrovači R3, proto se směrovače R2 i R1 od nového multicastového stromu odříznou zprávou JoinPrune.

Zprávy přijaté směrovači při simulaci:

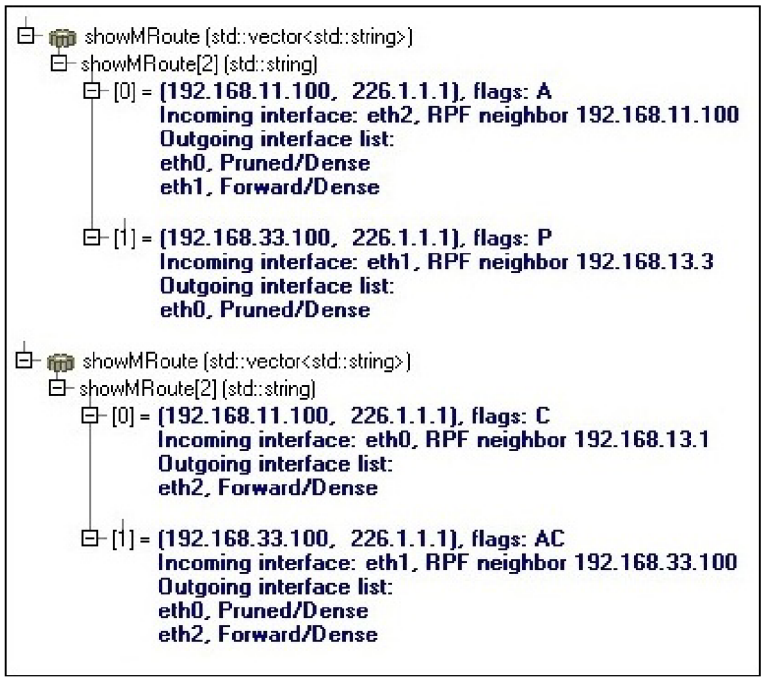
```
619.000044639996 R1(100): PIMJoinPrune (192.168.12.2, 224.0.0.13)
619.000051359995 R3(100): PIMJoinPrune (192.168.13.1, 224.0.0.13)
```

Zprávy zachycené na lince R1-R2 na reálné síti:

```
3076 619.339000   192.168.12.2        224.0.0.13      PIMv2    Join/Prune
```

Zprávy zachycené na lince R1-R3 na reálné síti:

```
3887 607.906000   192.168.13.1        224.0.0.13      PIMv2    Join/Prune
```



Obrázek 7.18: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) ze simulace (krok 9).

```
(192.168.11.100, 226.1.1.1), 00:02:17/00:02:50, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/0, Prune/Dense, 00:02:17/00:00:47
    FastEthernet0/1, Forward/Dense, 00:02:17/00:00:00

(192.168.33.100, 226.1.1.1), 00:00:33/00:02:30, flags: PT
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.3
  Outgoing interface list:
    FastEthernet0/0, Prune/Dense, 00:00:33/00:02:27
```

```
(192.168.11.100, 226.1.1.1), 00:02:30/00:02:53, flags: T
  Incoming interface: FastEthernet0/1, RPF nbr 192.168.13.1
  Outgoing interface list:
    FastEthernet0/3, Forward/Dense, 00:02:30/00:00:00

(192.168.33.100, 226.1.1.1), 00:00:46/00:02:53, flags: T
  Incoming interface: FastEthernet0/2, RPF nbr 0.0.0.0
  Outgoing interface list:
    FastEthernet0/1, Prune/Dense, 00:00:46/00:02:13
    FastEthernet0/3, Forward/Dense, 00:00:46/00:00:00
```

Obrázek 7.19: Multicastové směrovací tabulky směrovače R1 (nahoře) a R3 (dole) z reálné sítě (krok 9).

7.3 Shrnutí

V simulačním nástroji OMNeT++ a na reálné síti jsme si vyzkoušeli běžné situace, které mohou při směrování multicastu nastat. Zaznamenávali jsme v obou prostředích PIM zprávy

zasílané sítí a stavy multicastových směrovacích tabulek pro další porovnání.

Ze srovnání je zřejmé, že zprávy se při simulaci i v reálné síti zasílají stejné. Stav tabulek je téměř shodný. Drobné rozdíly pouze vycházejí z jiných příznaků uchovávaných u záznamů cest. Cisco příznaky totiž uchovává i u (*,G) cest, které se ale v naši multicastové směrovací tabulce nevytváří. Není pro to totiž žádný důvod, zbytečně bychom ukládali do tabulky něco, co zatím nevyužijeme. Některé příznaky nejsou pro simulaci naopak vůbec zásadní.

Tímto testem jsme si především ověřili, že implementace popsaná v kapitole 6 je funkční, správná a shodná s chováním Cisco směrovačů. Na přiloženém DVD (příloha B) může zvídavý čtenář nalézt další simulační příklady se složitějšími topologiemi. Jeden příklad simuluje i pád linky.

Závěr

Tato práce se zabývala problematikou multicastových směrovacích protokolů rodiny PIM se zaměřením na protokol PIM-DM. Seznámila nás také se simulačním nástrojem OMNeT++, knihovnami INET a ANSAINET a jejich možnostmi na poli modelování a simulace multicastového provozu. Cílem práce bylo rozšíření nástroje OMNeT++ o multicastové směrování. Navržené rozšíření bylo implementováno a jeho korektnost byla ověřena porovnáním simulace s reálnou sítí.

Vlastní přínos

Nejprve jsem se seznámila se simulačním nástrojem OMNeT++. Abych mohla jeho funkcionalitu v budoucnu rozšířit, prozkoumala jsme jeho aktuální možnosti modelování a simulování multicastu. Zjistila jsem, že samotný OMNeT++ ani knihovna INET nejsou dostačující, neboť multicast téměř nepodporují. Jediná podpora, v podobě protokolu IGMP, je součástí rozšiřující knihovny ANSAINET, která se vyvíjí v rámci fakultního projektu ANSA.

Následně jsem nastudovala fungování multicastového směrování. Obecné základy jsem rozšířila o poznatky týkající se protokolů PIM-DM, PIM-SM, PIM-SSM a BiDir-PIM. Všechny informace jsem čerpala z příslušných RFC. Zaměřila jsem se především na protokol PIM-DM, u kterého jsem si podrobně prostudovala všechny používané zprávy, časovače a konečné automaty řídící jeho logiku. Zvýšenou pozornost jsem věnovala i často využívané verzi PIM-SM.

Aby se výsledek práce přiblížil co nejvíce realitě, obeznámila jsem se s konfigurací multicastových směrovacích protokolů na směrovačích firmy Cisco. Konkrétně mě zajímala konfigurace vybraného PIM režimu, statické nastavení RP směrovačů, způsoby dynamické konfigurace RP směrovačů, monitorování a verifikace protokolu. Zběžně jsem se podívala i na konfiguraci méně využívaných protokolů PIM-SSM a BiDir-PIM.

Po nabytí nových poznatků jsem začala s návrhem rozšíření simulátoru. Nejprve jsem navrhla strukturu modelu PIM a jeho umístění v rámci modelu směrovače, který se nachází v knihovně ANSAINET. Navrhla jsem také rozšíření knihovny o multicastovou směrovací tabulku, tabulku rozhraní a tabulku sousednosti. Pokračovala jsme návrhem modelů PIM Splitter, který provádí úkony společné pro všechny protokoly PIM, a PIM-DM. Na základě konečných automatů z RFC jsem naplánovala posloupnost akcí, kterou budou oba modely periodicky provádět.

Navrhnout jsem také musela rozšíření konfiguračního souboru simulace o nepostradatelnou konfiguraci protokolu PIM. Rozšíření se týká jak protokolu PIM-DM, tak protokolu PIM-SM. Implementovala jsem rozšíření modulu Device Configurator, který nyní dokáže načíst nově navrhnutou strukturu konfiguračního souboru.

Pokračovala jsem implementací navrhnutých rozšíření. V jazyce NED jsem vytvořila

složený model protokolu PIM a integrovala jsem ho do modulu směrovače. V rámci rozšíření jsem dále naprogramovala mnoho důležitých tříd, které implementují multicastovou tabulku, tabulku rozhraní, tabulku sousednosti, PIM zprávy, PIM časovače, PIM Splitter a PIM-DM. Rozšířila jsem také implementaci síťové vrstvy tak, aby dokázala zpracovávat multicastová data a uměla spolupracovat s multicastovou tabulkou.

Implementaci jsem otestovala na několika simulacích s různými topologiemi simulující různé situace, které v síti mohou nastat. Zjišťovala jsem, zda se chování simulace shoduje s chováním popsaným v RFC. Následně jsem vybrala jednu topologii, kterou jsem sestavila i z reálných směrovačů Cisco, a porovnala jsem chování protokolu PIM v obou prostředích. Po srovnání odchycených dat ze simulace i reálné sítě jsem dospěla k závěru, že implementace byla provedena správně.

Výsledky, kterých jsem v rámci diplomové práce dosáhla, jsem prezentovala na konferenci a soutěži studentské tvůrčí činnosti STUDENT EEICT 2012. Porota ocenila můj příspěvek[17] udělením 2. místa v kategorii Infomační systémy.

Další vývoj

Třídy, které jsem naprogramovala, budou v průběhu léta společně s dalšími částmi knihovny ANSAINET, které vznikaly na naší fakultě, integrovány do knihovny INET a stanou se její nedílnou součástí. S některou z dalších verzí knihovny INET si tak budou moci zájemci z celého světa vyzkoušet simulaci multicastového směrovacího protokolu PIM v podobě, kterou jsem sama implementovala.

Touto integrací se zjednoduší i další rozšiřování protokolu, protože již nebudeme tolik závislí na změnách a rozdílech, které vznikají s každou novou verzí knihovny INET. Další možností rozšíření této práce je implementace protokolu PIM-SM. Tento protokol je výrazně složitější, ale případný autor bude mít práci lehčí díky mnou implementované obecné podpoře protokolu PIM (tabulky, PIM Splitter, atd.).

Důležité je také rozšíření knihovny INET o podporu multicastu na L2 vrstvě. Je potřeba implementovat používání multicastových MAC adres a schopnost přepínače správně zacházet s multicastovými daty. S tím souvisí také možná implementace technologie IGMP Snooping.

Možnosti dalšího rozšiřování je celá řada, jak na straně multicastového směrování (protokoly PIM-SM, PIM-SSM, BiDir-PIM), tak i na straně přihlašování do multicastové skupiny (IGMPv3). V dnešní době začíná být také aktuální používání protokolu IPv6. Proto bude žádoucí také implementace IPv6 multicastu a s tím souvisejících protokolů (MLD, PIM pro IPv6).

Literatura

[1] ANSAWiki [online]. 2012 [cit. 2012-01-05].
URL https://nes.fit.vutbr.cz/ansa/pmwiki.php

[2] INET Framework for OMNeT++. Manual [online]. 2012 [cit. 2012-01-05].
URL http://inet.omnetpp.org/doc/inet-manual-DRAFT.pdf

[3] Adams, A.; Nicholas, J.; Siadak, W.: Protocol Independent Multicast - Dense Mode (PIM-DM): Protocol Specification (Revised). RFC 3973 (Experimental), Leden 2005.
URL http://www.ietf.org/rfc/rfc3973.txt

[4] Bhaskar, N.; Gall, A.; Lingard, J.; aj.: Bootstrap Router (BSR) Mechanism for Protocol Independent Multicast (PIM). RFC 5059 (Proposed Standard), Leden 2008.
URL http://www.ietf.org/rfc/rfc5059.txt

[5] Bhattacharyya, S.: An Overview of Source-Specific Multicast (SSM). RFC 3569 (Informational), Červenec 2003.
URL http://www.ietf.org/rfc/rfc3569.txt

[6] Brent Stewart: *CCNP BSCI Official Exam Certification Guide, Fourth Edition*. Cisco Press, 2007, ISBN 1-58720-147-X.

[7] Cain, B.; Deering, S.; Kouvelas, I.; aj.: Internet Group Management Protocol, Version 3. RFC 3376 (Proposed Standard), Říjen 2002.
URL http://www.ietf.org/rfc/rfc3376.txt

[8] Cisco Systems, Inc.: *Cisco IOS IP Configuration Guide, Release 12.2 [online]*. 2001-2006 [cit. 2012-01-10].
URL http://www.cisco.com/en/US/docs/ios/12_2/ip/configuration/guide/1cfbook.pdf

[9] Cisco Systems, Inc.: *CCNP: Building Scalable Internetworks v5.0 - Lab*. 2006.

[10] Černý, M.: *Modelování IPv6 v prostředí OMNeT++*. Diplomová práce, FIT VUT v Brně, 2011.

[11] Fenner, B.; Handley, M.; Holbrook, H.; aj.: Protocol Independent Multicast - Sparse Mode (PIM-SM): Protocol Specification (Revised). RFC 4601 (Proposed Standard), Srpen 2006.
URL http://www.ietf.org/rfc/rfc4601.txt

[12] Fenner, W. C.: Internet Group Management Protocol, Version 2. RFC 2236 (Standard), Listopad 1997.
URL http://tools.ietf.org/html/rfc2236

[13] Handley, M.; Kouvelas, I.; Speakman, T.; aj.: Bidirectional Protocol Independent Multicast (BIDIR-PIM). RFC 5015 (Proposed Standard), Říjen 2007.
URL http://www.ietf.org/rfc/rfc5015.txt

[14] Mateleško, P.: *Simulování multicastových přenosů v simulátoru OMNeT++.* Bakalářská práce, FIT VUT v Brně, 2010.

[15] Meyer, D.; Lothberg, P.: GLOP Addressing in 233/8. RFC 3180 (Best Current Practice), Září 2001.
URL http://www.ietf.org/rfc/rfc3180.txt

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

[17] Rybová, V.: Multicast Routing Modelling in OMNeT++. In *Proceedings of the 18th Conference STUDENT EEICT 2012: Volume 2*, LITERA Brno, 2012, ISBN 978-80-214-4461-4, s. 193–195.
URL http://www.feec.vutbr.cz/EEICT/

[18] Scherfel, P.: *Simulace chování sítě na základě analýzy konfiguračních souborů aktivních síťových zařízení.* Bakalářská práce, FIT VUT v Brně, 2009.

[19] Thaler, D.; Fenner, B.; Quinn, B.: Socket Interface Extensions for Multicast Source Filters. RFC 3678 (Informational), Leden 2004.
URL http://www.ietf.org/rfc/rfc3678.txt

[20] Thaler, D.; Handley, M.; Estrin, D.: The Internet Multicast Address Allocation Architecture. RFC 2908 (Historic), Září 2000.
URL http://www.ietf.org/rfc/rfc2908.txt

[21] Varga, A.: *OMNeT++. User manual, version 4.1 [online].* 2010 [cit. 2012-01-03].
URL http://www.omnetpp.org/doc/omnetpp41/Manual.pdf

[22] Vida, R.; Costa, L.: Multicast Listener Discovery Version 2 (MLDv2) for IPv6. RFC 3810 (Proposed Standard), Červen 2004.
URL http://www.ietf.org/rfc/rfc3810.txt

[23] Welcher, P. J.: PIM Dense Mode [online]. 2001 [cit. 2012-04-18].
URL http://www.netcraftsmen.net/welcher/papers/multicast02.html

[24] Wikipedia: Internet Group Management Protocol [online]. 2012 [cit. 2012-01-05].
URL http://cs.wikipedia.org/wiki/Internet_Group_Management_Protocol

[25] Wikipedia: Reverse path forwarding [online]. 2012 [cit. 2012-01-05].
URL http://en.wikipedia.org/wiki/Reverse_path_forwarding

[26] Wikipedia: Type-length-value [online]. 2012 [cit. 2012-04-18].
URL http://en.wikipedia.org/wiki/Type-length-value

Příloha A

Seznam zkratek

ACL = Access Control List
ANSA = Automated Network-wide Security Analysis
ASM = Any-Source Multicast
BiDir-PIM = Bidirectional PIM
BSR = Bootstrap Router
DF = Designated Forwarder
DR = Designated Router
IGMP = Internet Group Management Protocol
IP = Internet Protocol
ISM = Internet Standard Multicast
OSPF = Open Shortest Path First
PIM = Protocol Independent Multicast
PIM-DM = PIM - Dense Mode
PIM-SM = PIM - Sparse Mode
PIM-SSM = PIM - Source Specific Mode
PMBR = PIM Multicast Border Router
PPP = Point-to-Point Protocol
MFIB = Multicast Forwarding Information Base
MLD = Multicast Listener Discovery
MPLS = Multiprotocol Label Switching
MRIB = Multicast Routing Information Base
MSDP = Multicast Source Discovery Protocol
RIP = Routing Information Protocol
RP = Rendezvous Point
RPA = Rendezvous Point Address
RPL = Rendezvous Point Link
RPF = Reverse Path Forwarding
SCTP = Stream Control Transmission Protocol
SSM = Source-Specific Multicast
STP = Spanning Tree Protocol
TCP = Transmission Control Protocol
TIB = Tree Information Base
UDP = User Datagram Protocol
VLAN = Virtual Local Area Network
XML = Extensible Markup Language

Příloha B

Obsah DVD

V následující tabulce B.1 je uveden obsah přiloženého DVD.

ANSAINET\	Rozbalená a přeložená knihovna ANSAINET i s upravenými a přidanými soubory a vlastními příklady simulací. V tomto stavu se využívala pro tuto práci.
doc\	Programová dokumentace ve formátu HTML vygenerovaná programem Doxygen.
examples\	Simulační příklady, na kterých byl odzkoušen protokol PIM-DM a ostatní komponenty. Ve složce FinalTest je simulace používaná v kapitole 7 pro srovnání s reálnou topologií. Příklady jsou také vloženy ve struktuře složky ANSAINET.
final test\	Simulační model z kapitoly 7 a výstupy simulace a reálné sítě.
impmenetation\	Třídy implementující protokol PIM-DM, multicastovou tabulku a síťovou vrstvu. Jsou také vloženy ve struktuře složky ANSAINET.
instalation\	Instalační soubory pro OMNeT++. Součástí je i instalační příručka.
tex\	Zdrojové soubory této práce psané v LaTeX.
doc.chm	Programová dokumentace ve formátu CHM vygenerovaná programem Doxygen.
projekt.pdf	Elektronická verze této práce v formátu PDF.
readme.txt	Obsah DVD.

Tabulka B.1: Obsah přiloženého DVD.

Příloha C

Implementace protokolu PIM

V této příloze uvádíme kompletní hlavičkové soubory k třídám PimSplitter a pimDM implementující protokol PIM (bez vložených knihoven apod.).

C.1 Třída PimSplitter

```
class PimSplitter : public cSimpleModule, protected INotifiable
{
private:
   IRoutingTable *rt;            /**< Pointer to routing table. */
   MulticastRoutingTable *mrt;   /**< Pointer to multicast routing table. */
   IInterfaceTable *ift;         /**< Pointer to interface table. */
   NotificationBoard *nb;        /**< Pointer to notification table. */
   PimInterfaceTable *pimIft;    /**< Pointer to table of PIM interfaces. */
   PimNeighborTable *pimNbt;     /**< Pointer to table of PIM neighbors. */
   const char *hostname;         /**< Router hostname. */

   void processPIMPkt(PIMPacket *pkt);
   void processNLTimer(PIMTimer *timer);

   // methods for Hello packets
   PIMHello* createHelloPkt(int iftID);
   void sendHelloPkt();
   void processHelloPkt(PIMPacket *pkt);

   // process notification
   void receiveChangeNotification(int category, const cPolymorphic *details);
   virtual void newMulticast(IPAddress destAddr, IPAddress srcAddr);
   void igmpChange(InterfaceEntry *interface);

   // not in use
   bool LoadConfigFromXML(const char *filename);

protected:
   virtual int numInitStages() const  {return 5;}
```

```
   virtual void handleMessage(cMessage *msg);
   virtual void initialize(int stage);

public:
   PimSplitter(){};
};
```

C.2 Třída pimDM

```
class pimDM : public cSimpleModule, protected INotifiable
{
private:
   IRoutingTable *rt;            /**< Pointer to routing table. */
   MulticastRoutingTable *mrt;   /**< Pointer to multicast routing table. */
   IInterfaceTable *ift;         /**< Pointer to interface table. */
   NotificationBoard *nb;        /**< Pointer to notification table. */
   PimInterfaceTable *pimIft;    /**< Pointer to table of PIM interfaces. */
   PimNeighborTable *pimNbt;     /**< Pointer to table of PIM neighbors. */

   // process events
   void receiveChangeNotification(int category, const cPolymorphic *details);
   void newMulticast(MulticastIPRoute *newRoute);
   void newMulticastAddr(addRemoveAddr *members);
   void oldMulticastAddr(addRemoveAddr *members);
   void dataOnPruned(IPAddress destAddr, IPAddress srcAddr);
   void dataOnNonRpf(IPAddress group, IPAddress source, int intId);
   void dataOnRpf(MulticastIPRoute *route);
   void rpfIntChange(MulticastIPRoute *route);

   // process timers
   void processPIMTimer(PIMTimer *timer);
   void processPruneTimer(PIMpt * timer);
   void processGraftRetryTimer(PIMgrt *timer);
   void processSourceActiveTimer(PIMsat * timer);
   void processStateRefreshTimer(PIMsrt * timer);

   // create timers
   PIMpt* createPruneTimer(IPAddress source, IPAddress group, int intId,
                           int holdTime);
   PIMgrt* createGraftRetryTimer(IPAddress source, IPAddress group);
   PIMsat* createSourceActiveTimer(IPAddress source, IPAddress group);
   PIMsrt* createStateRefreshTimer(IPAddress source, IPAddress group);

   // process PIM packets
   void processPIMPkt(PIMPacket *pkt);
   void processJoinPruneGraftPacket(PIMJoinPrune *pkt, PIMPacketType type);
   void processPrunePacket(MulticastIPRoute *route, int intId, int holdTime);
```

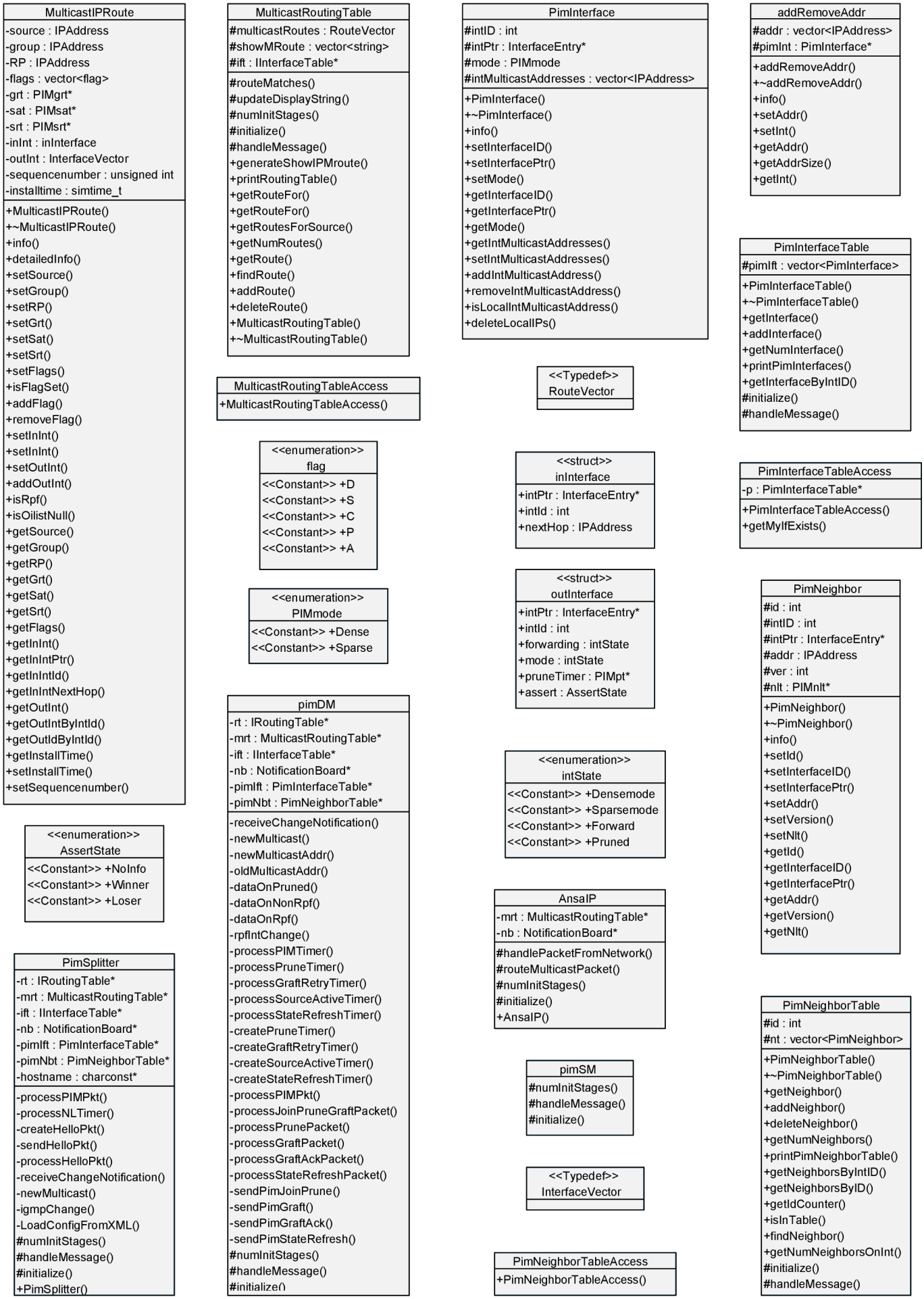
```
    void processGraftPacket(IPAddress source, IPAddress group,
                            IPAddress sender, int intId);
    void processGraftAckPacket(MulticastIPRoute *route);
    void processStateRefreshPacket(PIMStateRefresh *pkt);

    //create PIM packets
    void sendPimJoinPrune(IPAddress nextHop, IPAddress src, IPAddress grp,
                          int intId);
    void sendPimGraft(IPAddress nextHop, IPAddress src, IPAddress grp,
                      int intId);
    void sendPimGraftAck(PIMGraftAck *msg);
    void sendPimStateRefresh(IPAddress originator, IPAddress src,
                            IPAddress grp, int intId, bool P);

protected:
    virtual int numInitStages() const  {return 5;}
    virtual void handleMessage(cMessage *msg);
    virtual void initialize(int stage);
};
```

C.3 Diagram tříd

Na obrázku C.1 můžete vidět diagram všech tříd, které v rámci této práce vznikly.



Obrázek C.1: Diagram tříd, které vznikli v rámci této práce.