VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

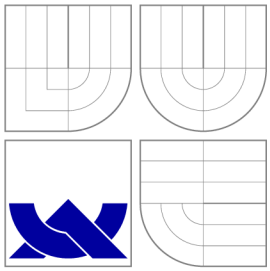
MODELOVÁNÍ SMĚROVACÍHO PROTOKOLU EIGRP

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

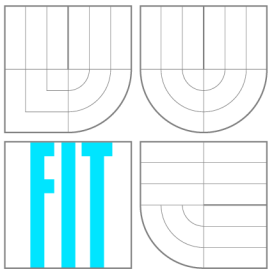
AUTOR PRÁCE
AUTHOR

Bc. JAN BLOUDÍČEK

BRNO 2014



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

MODELOVÁNÍ SMĚROVACÍHO PROTOKOLU EIGRP

MODELLING OF EIGRP ROUTING PROTOCOL

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

Bc. JAN BLOUDÍČEK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. VLADIMÍR VESELÝ

BRNO 2014

Abstrakt

Simulace v oblasti počítačových sítí umožňuje analýzu chování sítě a nakonfigurovaných protokolů. Tato práce se zaměřuje na směrovací protokol EIGRP a jeho začlenění do simulačního nástroje OMNeT++. Součástí textu je detailní popis tohoto protokolu a jeho konfigurace na zařízení značky Cisco. Jsou zde shrnuty principy simulátoru OMNeT++ a knihovny INET. Dále je nastíněn návrh rozšíření nástroje OMNeT++ o tento směrovací protokol. Následuje implementace protokolu podle provedených návrhů. Nakonec je implementované řešení porovnáno s výstupem reálných zařízení.

Abstract

The network simulation allows analysis of the computer networks behavior and configured protocols. This thesis focuses on the EIGRP routing protocol and its integration into the OMNeT++ simulation enviroment. The text includes a detailed description of the protocol and its configuration on Cisco devices. Furthermore, the text focuses on design of extension that supports routing protocol. The following describes implementation of the protocol according to design. Finally, the implemented solution is compared with the output of real devices.

Klíčová slova

Simulace sítí, EIGRP, směrovací protokol, Cisco, OMNeT++, INET, ANSA.

Keywords

Network Simulation, EIGRP, routing protocol, Cisco, OMNeT++, INET, ANSA.

Citace

Jan Bloudíček: Modelování směrovacího protokolu EIGRP, diplomová práce, Brno, FIT VUT v Brně, 2014

Modelování směrovacího protokolu EIGRP

Prohlášení

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

.....
Jan Bloudíček
12. června 2014

Poděkování

Chtěl bych poděkovat Ing. Vladimíru Veselému za odbornou pomoc a čas věnovaný konzultacím. Jako odměnu mu rád věnuji recept na avokádovou polévku. Na její přípravu je třeba jeden litr masového vývaru, jedno velké nebo dvě menší avokáda, která musí být měkká. Dále 200 ml smetany a sůl. Do vroucího vývaru přidáme oloupané a rozmixované avokádo. Polévku znovu přivedeme k varu a odstavíme z plotny. Nakonec přidáme smetanu a osolíme dle chuti. K polévce můžeme podávat na kostky nakrájenou a opraženou housku.

Obsah

1 Úvod **3**

2 Enhanced Interior Gateway Routing Protocol **4**
2.1 EIGRP zprávy . . . 4
2.1.1 Hlavička zprávy . . . 5
2.1.2 Užitečný obsah zprávy . . . 6
2.2 Vztah sousedství . . . 7
2.2.1 Tabulka sousedů . . . 8
2.2.2 Vytvoření vztahu sousedství . . . 9
2.2.3 Udržování vztahu sousedství . . . 10
2.3 Kompozitní metrika . . . 10
2.3.1 Výpočet klasické metriky . . . 11
2.4 Rozšířená metrika . . . 11
2.4.1 Výpočet rozšířené metriky . . . 12
2.5 Správa cest . . . 12
2.5.1 Tabulka topologie . . . 13
2.5.2 Stub směrování . . . 13
2.6 Diffusing Update Algorithm . . . 14
2.6.1 Popis algoritmu . . . 15
2.6.2 Pravidla pro omezení smyček . . . 19
2.7 Reliable Transport Protocol . . . 19
2.8 Vyvažování zátěže . . . 20

3 Konfigurace EIGRP na Cisco zařízeních **21**
3.1 Konfigurace pro IPv4 . . . 21
3.2 Konfigurace pro IPv6 . . . 23
3.3 Pojmenovaná konfigurace . . . 23
3.4 Ověření konfigurace . . . 24

4 OMNeT++ a knihovna INET **26**
4.1 INET . . . 27
4.2 Připravenost pro EIGRP . . . 27

5 Návrh EIGRP protokolu **28**
5.1 EIGRP proces . . . 28
5.2 Abstraktní datové typy . . . 29
5.3 Návrh PDM pro IPv4 . . . 30
5.3.1 Zpracování událostí . . . 30

5.3.2 Uplatnění pravidel pro omezení smyček . . . 32
5.3.3 Stub směrování . . . 32
5.3.4 Správa cest . . . 33
5.4 Návrh RTP . . . 33
5.5 Návrh konfigurace EIGRP . . . 34
5.5.1 Konfigurace na úrovni směrovacího procesu . . . 34
5.5.2 Konfigurace na úrovni rozhraní . . . 35

6 Implementace EIGRP v OMNeT++ 36
6.1 Modul EIGRP . . . 36
6.2 Třídy pro uchování dat . . . 37
6.2.1 Tabulka rozhraní EigrpInterfaceTable . . . 37
6.2.2 Tabulka sousedů EigrpIpv4NeighborTable . . . 38
6.2.3 Tabulka topologie EigrpIpv4TopologyTable . . . 38
6.2.4 EIGRP zprávy . . . 39
6.2.5 EIGRP časovače . . . 40
6.3 Modul EigrpIpv4Pdm . . . 40
6.3.1 Zpracování notifikací . . . 40
6.4 Modul EigrpRtp . . . 41
6.5 Konfigurace EIGRP modulu . . . 42

7 Porovnání simulace a reálné sítě 43
7.1 Test ustavení sousedství . . . 44
7.2 Test pádu rozhraní . . . 45
7.3 Test zapnutí rozhraní . . . 47
7.4 Test vypnutí Split Horizon . . . 48
7.5 Test nestejnoměrného vyvažování zátěže . . . 50
7.6 Test vypnutí vyvažování zátěže . . . 50
7.7 Test stub směrování . . . 51

8 Závěr 53

A Obsah CD 57

B Verze EIGRP 58

C Diagram tříd 59

Kapitola 1

Úvod

Internet je celosvětově rozšířená komunikační síť, mající čím dál větší vliv na životy lidí. Počet zařízení schopných komunikovat po síti se stále zvyšuje a s tím souvisí i vyšší nároky na její správu a efektivitu. Pro uspokojení těchto potřeb vznikl nespočet technologií a jednou z nich jsou směrovací protokoly. Šíří v síti směrovací informace a umožňují pružnou reakci na změny do značné míry bez účasti člověka.

Nasazení směrovacího protokolu do rozlehlé sítě může přinést problémy a případně způsobit její nedostupnost pro koncové uživatele. Výhodnější je ověřit správnost konfigurace mimo reálnou síť. Vhodným řešením je použít simulátor, jenž umožňuje vyhodnotit model sítě opakovaně s různými parametry. Lze také velmi dobře analyzovat chování protokolů a komunikaci mezi zařízeními. Simulační nástroj OMNeT++ takové funkce nabízí.

Práce je součástí projektu ANSA a zabývá se směrovacím protokolem Enhanced Interior Gateway Routing Protocol a jeho včleněním do OMNeT++. Ve druhé kapitole tohoto dokumentu je popsán zmiňovaný směrovací protokol, jeho vlastnosti a chování. Ve třetí kapitole se zabývám konfigurací EIGRP na zařízeních od firmy Cisco. Čtvrtá kapitola popisuje principy nástroje OMNeT++ a knihovny INET a možnosti pro nasazení EIGRP do jejich architektury. Kapitola pátá obsahuje návrh modulu EIGRP a jeho včlenění do stávající struktury OMNeT++. Na základě návrhu byl tento protokol implementován a kapitola sedmá popisuje způsob řešení podstatných částí. V osmé kapitole jsou uvedeny výsledky testů, jejichž cílem je ověřit správnost implementace protokolu vůči chování reálných zařízení. Závěrečná kapitola shrnuje přínos této práce a možné směry pokračování.

Kapitola 2

Enhanced Interior Gateway Routing Protocol

Směrovací protokoly jsou určeny pro šíření směrovacích informací mezi směrovači. Tento způsob konfigurace se nazývá dynamické směrování a představuje alternativu ke statickému směrování, kdy se cesty do směrovacích tabulek vkládají ručně. Mezi výhody dynamického směrování patří např. pružná reakce na změny v síti, méně náročná konfigurace a lepší spravovatelnost.

EIGRP [20] je proprietární směrovací protokol od firmy Cisco a nahrazuje starší protokol Internet Gateway Routing Protocol (IGRP) [15]. Náleží do kategorie protokolů Interior Gateway Protocol určených pro šíření směrovacích informací uvnitř autonomního systému.

Patří do skupiny Distance Vector směrovacích protokolů, které pro rozhodování o směrování zpráv využívají vzdálenost a směr do cíle. Vyznačují se také periodickým odesíláním celé směrovací tabulky sousedním směrovačům. Mezi Distance Vector protokoly se dále řadí například Routing Information Protocol (RIP) a IGRP. EIGRP má některé vlastnosti z Link State protokolů. Směrovací informace šíří sítí pouze při výskytu události a přenáší se změněná data, ne celá směrovací tabulka. Jelikož má vlastnosti z Distance Vector i Link State protokolů, nazývá se Advanced Distance Vector protokol.

Je to classless protokol, protože ve zprávách s informacemi o cestě přenáší i masku sítě. Lze jej však nakonfigurovat tak, že se bude chovat jako classful protokol. Podporuje tedy Variable Length Subnet Mask (VLSM) a Classless Interdomain Routing (CIDR).

Pracuje na síťové vrstvě ISO/OSI modelu a je nezávislý na použitém protokolu. Podporuje IPv4, IPv6, AppleTalk a IPX (Novell NetWare Internetwork Exchange). Používá Protocol Dependent Modules (PDM), které se starají o podporu těchto protokolů. Každý protokol na síťové vrstvě má svůj modul. Další text bude zaměřen především na síťový protokol IPv4 a IPv6. EIGRP podporuje autentizaci, také je možné využít sumarizaci cest, a to manuální i automatickou. Dnes jej podporují pouze zařízení značky Cisco.

Dále v textu se bude objevovat termín EIGRP autonomní systém (AS), který představuje skupinu směrovačů používajících protokol EIGRP se stejným číslem AS. Tento pojem není shodný s AS používaným v Border Gateway Protocol (BGP).

2.1 EIGRP zprávy

Protokol EIGRP používá zprávy pro šíření směrovacích informací v síti a také pro vlastní režii. Zprávy jsou zapouzdřeny do paketů použitého síťového protokolu a jsou přenášeny

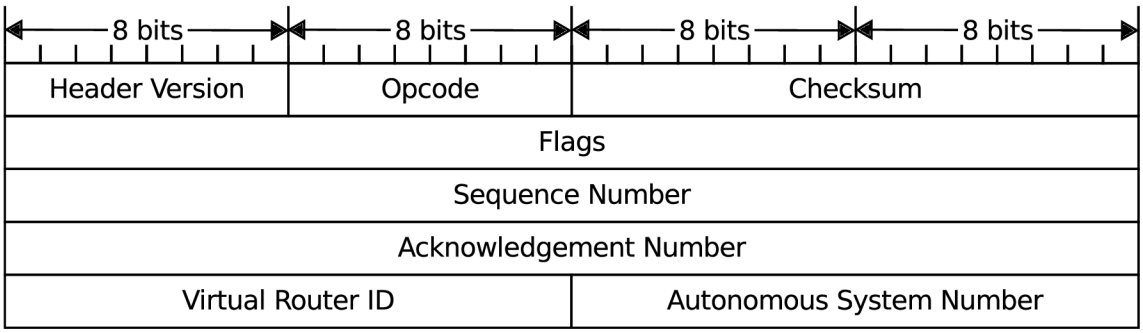
jako unicast nebo multicast. Pro multicastové zprávy se používá IPv4 adresa 224.0.0.10 nebo IPv6 adresa FF02::A [21].

Následuje přehled typů EIGRP zpráv, jejich vlastnosti a použití [21]. Za názvem zprávy je v závorce uveden její kód.

- **Hello** (0x05) - slouží především pro objevení sousedních směrovačů a udržování vztahu sousedství. Zpráva je obvykle adresována jako multicast a není vyžadováno potvrzení o jejím přijetí;
- **Ack** (0x05) - má význam potvrzení o přijetí zprávy a používá se pro spolehlivý přenos. Jedná se o unicastovou zprávu, která není nikdy potvrzována;
- **Update** (0x01) - zpráva přenáší směrovací informace, a proto je vždy potvrzovaná. Je adresována jako unicast nebo multicast;
- **Query** (0x03) - tuto zprávu využívá algoritmus DUAL pro difúzní výpočet. Obvykle je odesílána na multicastovou adresu a vyžaduje spolehlivý přenos;
- **Reply** (0x04) - je odpovědí na zprávu *Query* a jedná se o unicastovou zprávu přenášenou spolehlivým způsobem;
- **SIA-Query** (0x0A) a **SIA-Reply** (0x0B) - slouží jako prevence proti Stuck-In-Active (SIA) [12], jsou potvrzovány;
- **Request** (0x02) - žádost o specifické informace od sousedů. Přenášeny nespolehlivě jako unicast nebo multicast.

2.1.1 Hlavička zprávy

Zprávy jsou složeny z hlavičky a užitečného obsahu. Hlavička je pro všechny EIGRP zprávy stejná, její struktura je na obrázku 2.1. Následuje popis jednotlivých polí hlavičky [21].



Obrázek 2.1: Hlavička EIGRP zprávy

Header Version je verze hlavičky EIGRP zprávy, v současnosti se používá verze 2.

Opcode obsahuje kód zprávy určující její typ.

Checksum celé EIGRP zprávy. Je to běžný jedničkový doplněk stejný jako v IP popsaný v RFC 1071.

Flags je pole příznaků, které indikuje speciální význam zprávy. Jejich stručný popis je v tabulce 2.1.

Sequence Number je číslo zprávy pro zajištění spolehlivého přenosu.

Acknowledgement Number je číslo, které slouží pro potvrzení přijaté zprávy.

Virtual Router ID (VRID) je identifikátor směrovače.

Autonomous System Number je číslo autonomního systému.

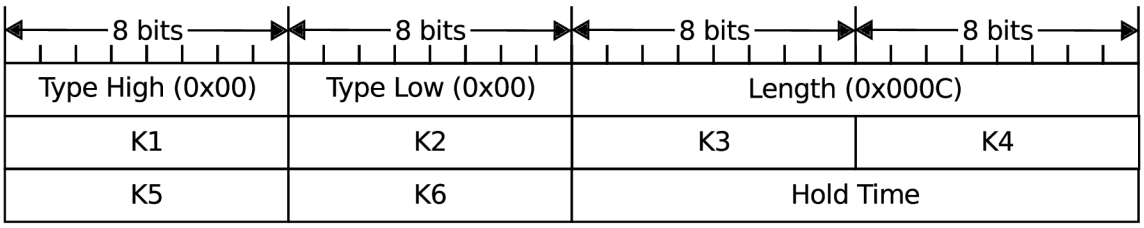
Název	Kód	Popis
INIT	1	Požadavek na zaslání obsahu směrovací tabulky. Používá se především při ustavení sousedství.
CR	2	Zprávu s tímto příznakem smí zpracovat jen směrovač, který je v Conditionally Receive módu, viz kapitolu 2.7.
RS	4	Zpráva s tímto příznakem oznamuje restart směrovače. Využívá jej funkce Nonstop Forwarding [9].
EOT	8	Příznakem End Of Table směrovač oznamuje, že odeslal veškeré cesty ze směrovací tabulky.

Tabulka 2.1: Pole s příznaky (Flags) v hlavičce

2.1.2 Užitečný obsah zprávy

Užitečných obsah zprávy [21] se skládá z proměnného počtu položek, které jsou ve formátu TLV (Type Length Value). Tento formát dovoluje rozšířit protokol o další funkce, aniž by se musely provádět změny v protokolu nebo v části zpracování paketů.

Type se dělí na dvě části. Type High určuje síťový protokol šířený pomocí EIGRP, Type Low identifikuje přenášenou TLV položku. Tabulku s možnými hodnotami tohoto pole lze nalézt v RFC [21, kap. A.7]. Length je velikost celé TLV položky v bytech. Value je užitečný obsah TLV položky proměnlivé velikosti. Dále je uveden popis TLV podstatných pro tuto práci.



Obrázek 2.2: TLV položka Parameter Type

Parameter Type je na obr. 2.2. Vyskytuje v *Hello* zprávách a obsahuje K-hodnoty pro výpočet metriky a časovač *Hold*. Podle RFC [21, kap. A.7.1] je TLV obsaženo i ve zprávě *Update* odesílané při ustavení sousedství mezi směrovači, což ovšem neodpovídá Cisco implementaci.

Stub routing TLV je určeno pro přenos informací o nastavení Stub funkce. Type má hodnotu 0x0006 a Length je vždy 6. Value obsahuje pole příznaků (2 Byty), které určují, jaké cesty bude Stub oznamovat ostatním. Příznaky jsou uvedeny v tabulce 2.2.

Příznak	Kód	Význam
Connected	0x01	Přímo připojené cesty.
Static	0x02	Staticky nakonfigurované cesty.
Summary	0x04	Sumarizované cesty.
Receive-Only	0x08	Stub nešíří žádné cesty.
Redistributed	0x10	Cesty získané z jiných zdrojů redistribucí.
Leak-Map	0x20	Selektivní výběr cest na základě definovaných pravidel.

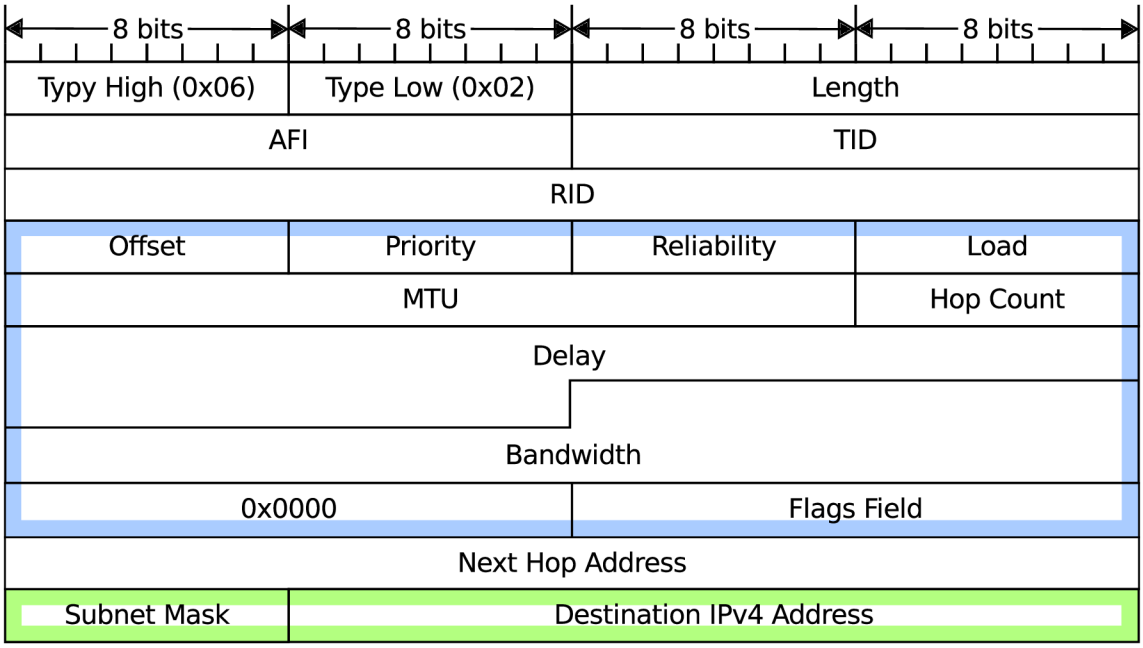
Tabulka 2.2: Příznaky pro stub směrování

Multi-Protocol Internal Type přenáší informace o interní cestě a s ní spojené metrice. Je určeno pro přenos IPv4 a IPv6 cest. Struktura TLV je na obrázku 2.3. Význam polí je následující:

- Address Family Identifier (AFI) je číslo určující typ přenášené síťové adresy;
- Topology Identifier (TID) je identifikátor logické topologie, který využívá funkce Multi-Topology Routing. Ve výchozím nastavení se používá základní topologie s hodnotou 0 [13];
- Router Identifier (RID) je identifikátor směrovače v syntaxi IPv4 adresy, který přenášenou cestu vytvořil;
- V bloku s rozšířenou metrikou (zvýrazněn modře) jsou uloženy hodnoty pro výpočet metriky, jejichž popis je v kapitole 2.3. Dále je tu Offset, jenž souvisí s možností vložit do TLV další položky [21, kap. A.9.2]. V MTU je obsažena nejnižší hodnota na cestě mezi směrovačem a cílem. Hop Count určuje počet směrovačů na cestě do cíle. Flags Field je pole příznaků, jenž definují stav cesty. Jejich význam lze najít v RFC [21, kap. A.8.1];
- Next Hop Address je adresa následujícího směrovače na cestě do cíle;
- V dalším bloku (zvýrazněn zeleně) je adresa a maska cesty.

2.2 Vztah sousedství

Vztah sousedství [24] mezi dvěma směrovači lze chápat jako stav, kdy oba znají potřebné informace od svého protějšku a pravidelně kontrolují jeho živost. Sousedství lze ustavit mezi směrovači, které jsou přímo propojené, a jejich konfigurace jim dovoluje si mezi sebou vyměňovat směrovací informace. Pro svoji potřebu protokol používá vždy primární IPv4 adresy, tzn. i pro vztah sousedství. EIGRP pro IPv6 využívá ke své činnosti link-local adresy. Aby mohl vznikout mezi směrovači tento vztah, musí být splněny následující podmínky [21]:



Obrázek 2.3: TLV položka pro IPv4 adresu interní cesty

- Sousedící směrovače mají na rozhraní, přes které se spolu snaží ustavit sousedství, povolené EIGRP;
- Primární IPv4 adresy obou směrovačů jsou ve stejné síti. V IPv6 takový požadavek není, zkontroluje se pouze, zda směrovače používají platné link-local adresy;
- Musí souhlasit čísla autonomního systému;
- Oba směrovače mají stejné všechny K-hodnoty pro výpočet metriky. Nesoulad kterékoli z nich vede k ukončení vztahu.

2.2.1 Tabulka sousedů

Je to tabulka všech přímo připojených směrovačů, se kterými je ustaven vztah sousedství. Pro každý nakonfigurovaný L3 protokol je udržována samostatná tabulka sousedů. Tabulka obsahuje následující informace [24]:

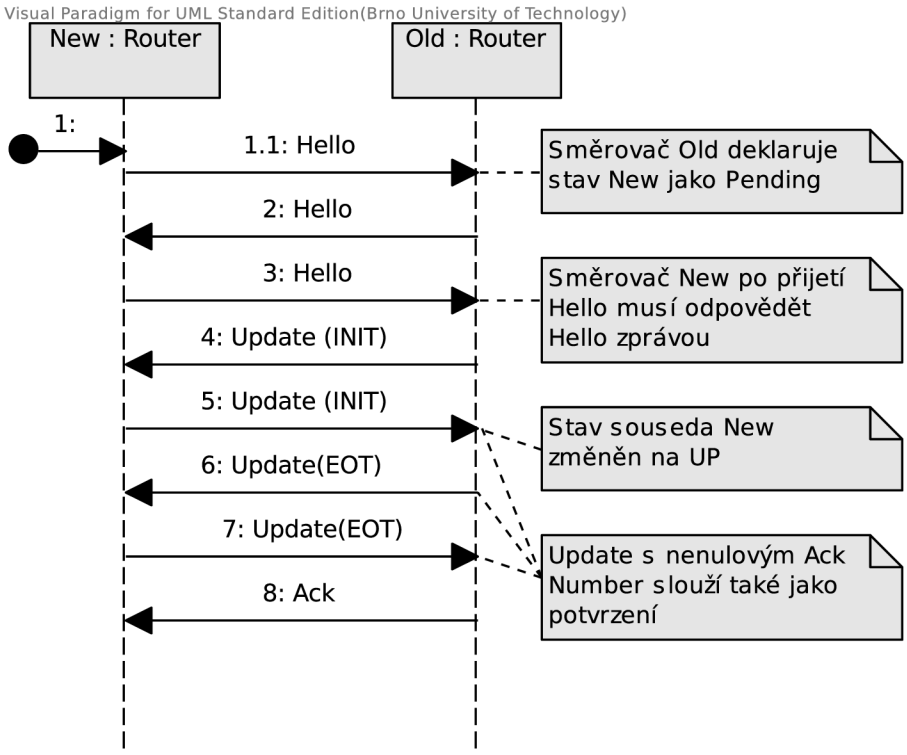
- **Interface** je rozhraní, na kterém je soused připojen;
- **Address** je IP adresa souseda;
- **Hold Time** je hodnota časovače *Hold* v sekundách;
- **Uptime** je doba uplynulá od ustavení sousedství;
- **Smooth Round Trip Timer (SRTT)** je průměrná doba v milisekundách, kterou zabere odeslat zprávu a přijmout potvrzení od souseda;

- **Retransmission Timeout (RTO)** je doba, kterou směrovač čeká na potvrzení na odeslanou zprávu;
- **Queue Count** vyjařuje počet EIGRP zpráv čekajících ve frontě na odeslání;
- **Sequence Number** obsahuje sekvenční číslo z poslední zprávy od souseda;

2.2.2 Vytvoření vztahu sousedství

Posloupnost zpráv přenášených při navázání sousedství je na obrázku 2.4 [21]. Přijme-li směrovač multicastovou zprávu *Hello* od souseda, o kterém dosud neměl žádný záznam v tabulce sousedů, odpoví mu zprávou *Hello*. Oba si na základě přijatých zpráv vytvoří záznam ve své tabulce sousedů. V dalším kroku si směrovače vzájemně pošlou zprávu *Update* na unicastovou adresu bez směrovacíh dat s nastaveným příznakem INIT. Následně si přepošlou potvrzení na přijaté zprávy *Update*. Tímto je ověřena unicastová i multicastová komunikace a sousedství je tedy ustaveno. Příjemce zprávy podle příznaku INIT zjistí, že by měl přeposlat všechny informace ze směrovací tabulky. Tyto informace jsou součástí následujících zpráv *Update*. Poslední zpráva *Update* má příznak End Of Table (odeslány všechny cesty ze směrovací tabulky). Informace z *Update* zpráv jsou uloženy do tabulky topologie a případně do směrovací tabulky.

Směrovač, který již má navázané sousedství, je ve stavu Up. Směrovač je ve stavu Pending, pokud zahájil proces ustavení sousedství a ještě ho nedokončil. Platí, že se nesmí odesílat *Update* a *Query* směrovači, který je ve stavu Pending.



Obrázek 2.4: Sled zpráv při ustavení sousedství mezi směrovači

2.2.3 Udržování vztahu sousedství

Směrovač pravidelně kontroluje stav svých sousedů pomocí periodického posílání zpráv *Hello*. EIGRP pro zajištění této funkce používá následující časovače [24]:

- **Hello Timer** určuje, jak často budou odesílány *Hello* zprávy. Hodnota závisí na typu linky, přes kterou se *Hello* zprávy přenáší. Ve výchozím nastavení je to 60 sekund pro Non-broadcast Multi Access rozhraní s rychlostí T1 (1,544 Mbps) a nižší. Na linkách s vyšší šířkou pásma je výchozí hodnota 5 sekund;
- **Hold Timer** je časovač pro udržování vztahu sousedství. Pokud směrovač nepřijme od souseda do této doby zprávu, pak jej označí za nedostupného a ukončí s ním vztah sousedství. Pro reset časovače vyhovuje jakákoli EIGRP zpráva. Výchozí hodnota časovače je trojnásobek časovače *Hello*.

Proces ukončení vztahu sousedství zahrnuje následující akce. Záznam o sousedovi je smazán z tabulky sousedů, všechny položky v tabulce topologie, které procházejí přes souseda, jsou smazány. Pokud jsou ve směrovací tabulce EIGRP cesty vedoucí přes tohoto souseda, pak jsou rovněž odebrány.

Zpráva Goodbye [24] je způsob, jak sousední směrovač informovat o ukončení vztahu sousedství. Jedná se o zprávu *Hello* s K-hodnotami K1 až K6 nastavenými na maximum. Goodbye zpráva se odesílá při ukončení sousedství nebo při neúspěšném vytvoření vztahu sousedství z důvodu nesouladu K-hodnot.

Tato zpráva se používá i pro funkci Graceful Shutdown [24], což je způsob, jak sdělit ostatním směrovačům v síti, že došlo k ukončení EIGRP procesu. Zrychluje konvergenci sítě, jelikož směrovače nemusí čekat na vypršení časovače *Hold*.

2.3 Kompozitní metrika

EIGRP používá pro ohodnocení cest 32 bitové číslo zvané metrika [21]. Základní metrika se skládá ze čtyř parametrů spojených s K-hodnotami, viz tabulku 2.3. K-hodnota je číslo, které může nabývat 0 (asociovaný parametr vyloučen z výpočtu metriky) až 255 (parametr má nejvyšší možnou váhu).

Parametr	Hodnota	Popis
Šířka pásma (Bandwidth)	K1 = 1	Je to statická hodnota nastavená na rozhraní. Výchozí hodnota určena typem linky (max. teoretická rychlost).
Zatížení (Load)	K2 = 0	Vyjadřuje zatížení linky, počítá se dynamicky na základě rychlosti odesílání paketů a šířky pásma.
Zpoždění (Delay)	K3 = 1	Je čas potřebný k přenesení jednobitové zprávy sousedovi. Statická hodnota.
Spolehlivost (Reliability)	K4 = 0, K5 = 0	Dynamická hodnota, která určuje pravděpodobnost, že zpráva bude úspěšně odeslána nebo přijata.

Tabulka 2.3: Hodnoty a parametry pro výpočet metriky

Výchozí hodnoty pro šířku pásma a zpoždění podle typu linky jsou uvedeny v RFC [21, str. 34]. Zatížení je dynamický parametr a využívá výstupní rychlost na rozhraní vyjádřenou jako exponenciální průměr rychlosti odesílání v b/s v časovém rozmezí pěti minut

ve výchozím nastavení. Hodnota zatížení je upravována každých pět sekund. Spolehlivost je také počítána dynamicky jako exponenciální průměr výskytu chyb při odesílání nebo přijímání zpráv v časovém rozmezí pěti minut. Počet chyb je pak vydělen počtem všech odeslaných paketů na rozhraní. Opět je aktualizována každých pět sekund [4]. MTU (Maximum Transmission Unit) se také přenáší v EIGRP zprávách, avšak při výpočtu metriky se nepoužívá.

2.3.1 Výpočet klasické metriky

Následuje popis výpočtu klasické metriky [21]. EIGRP metrika je zpětně kompatibilní s metrikou v IGRP, která je však 24 bitová. Proto se ve vzorcích pro výpočet šířky pásma a zpoždění násobí číslem 256. Cisco směrovače neprovádí výpočet v plovoucí čárce, při každém mezivýpočtu je tedy nutné provést zaokrouhlení dolů.

Pro výpočet metriky se uvažují hodnoty ze všech směrovačů na cestě do cíle, a to podle následujících pravidel:

- **Šířka pásma** - pro výpočet metriky se použije nejnižší šířka pásma ($BwMin$) v Kb/s ze všech rozhraní na cestě od směrovače do cílové sítě. Před dosazením se upraví dle vzorce 2.1;

$$Bw = 10^7 / BwMin * 256 \tag{2.1}$$

- **Zatížení** - bere se nejvíce zatížená linka mezi směrovačem a cílovou sítí. Vyjadřuje se procentuálně jako X/255, kde X je hodnota od 1 do 255.

- **Zpoždění** - do vzorce metriky se dosazuje součet zpoždění ($DlySum$) v mikrosekundách všech výstupních rozhraní na cestě od směrovače do cílové sítě. Před výpočtem metriky se upraví podle 2.2;

$$Dly = DlySum/10 * 256 \tag{2.2}$$

- **Spolehlivost** - použije se hodnota nejméně spolehlivé linky ze všech, které jsou na dané cestě mezi směrovačem a cílem. Vyjadřuje se procentuálně jako X/255.

Výpočet metriky pro K5 = 0 se řídí vzorcem 2.3. Pokud je hodnota K5 nenulová, pak dojde k úpravě, viz 2.4.

$$Metric = K1 * Bw + K2 * Bw/(256 - Load) + K3 * Dly \tag{2.3}$$

$$Metric = Metric * K5/(Reliability + K4) \tag{2.4}$$

2.4 Rozšířená metrika

Klasická metrika nevyhovuje při použití linek s šířkou pásma 1Gb/s a vyšší. Verze EIGRP 8.0 a pozdější využívají kromě klasické metriky i metriku rozšířenou [14]. Toto rozšíření umožňuje přidání dalších parametrů do výpočtu. Dále rozšířená metrika upravuje vzorce pro výpočet, nyní je výsledkem 64 bitové číslo.

Vzorec pro rozšířenou metriku není použit vždy. Uplatní se, jen když je protokol nastaven pomocí pojmenované konfigurace (Named Configuration), místo konfigurace autonomního systému (Autonomous System Configuration). Jinak se využije původní vzorec klasické metriky [11].

Při používání rozšířené metriky je nutné se vyrovnat s problémem uložení metriky do směrovací tabulky, kde je vyžadováno 32 bitové číslo. Řešením je škálování metriky pomocí faktoru, jehož výchozí hodnota je 128. Metrika se 64 bity se tedy podělí tímto číslem a teprve pak je vložena do směrovací tabulky [14].

Rozšířená metrika umožňuje do výpočtu přidat další dva parametry spojené s hodnotou K6, která je ve výchozím nastavení rovna 0. Jedná se o tyto parametry:

- **Směrodatná odchylka zpoždění** (Jitter) - použije se akumulovaná hodnota v mikrosekundách na cestě od směrovače do cílové sítě;
- **Spotřeba energie na rozhraní** (Energy) - sumarizovaná hodnota na cestě k cíli, je vyjádřena jednotkou Watt za přenesený Kb na daném rozhraní.

2.4.1 Výpočet rozšířené metriky

Následuje popis výpočtu rozšířené metriky. Nejdříve se vybere nejnižší šířka pásma $BwMin$ v Kb/s ze všech rozhraní na cestě od směrovače do cílové sítě. Hodnota je vkládána do TLV cesty do bloku s rozšířenou metrikou.

Dále se sečtou zpoždění v mikrosekundách všech výstupních rozhraní na cestě od směrovače do cílové sítě, výsledkem je $DlySum$. Hodnota je pak upravena pomocí vzorce 2.5, pokud je šířka pásma $BwMin$ vyšší než 1 Gb/s. Jinak se pro úpravu použije 2.6. Výsledkem je zpoždění v pikosekundách. Tato hodnota se vkládá do TLV cesty.

$$Dly = DlySum * 10^6 \tag{2.5}$$

$$Dly = (10^7 * 10^6)/BwMin \tag{2.6}$$

Následně se upraví šířka pásma pomocí 2.7 a dostaneme mezivýsledek označovaný jako *throughput* (Thr). Zpoždění je také upraveno na *latency* (Lat), a to vzorcem 2.8. Nově přidané parametry se jednoduše sečtou, viz 2.9.

$$Thr = (10^7 * 65536)/BwMin \tag{2.7}$$

$$Lat = (Dly * 65536)/10^6 \tag{2.8}$$

$$Ext = Energy + Jitter \tag{2.9}$$

Metrika je vypočtena vzorcem 2.10. Pokud je hodnota K5 různá od nuly, pak je nutné metriku ještě upravit pomocí 2.11.

$$Metric = K1 * Thr + K2 * Thr/(256 - Load) + K3 * Lat + K6 * Ext \tag{2.10}$$

$$Metric = Metric * K5/(Reliability + K4) \tag{2.11}$$

2.5 Správa cest

Cesta je strukturovaná hodnota složená z cílové sítě a adresy souseda (Next Hop). Rozlišují se dva typy cest, a to interní a externí. Interní cesty jsou uvnitř jednoho EIGRP autonomního systému (AS). Externí cesty jsou převzaté z jiného zdroje, což mohou být např. statické cesty ze směrovací tabulky, cesty z jiného směrovacího protokolu či jiného AS.

Každá cesta má podle svého zdroje přiřazenou hodnotu Administrative Distance (AD). Když je k dispozici více cest z různých zdrojů do stejné cílové sítě, vloží se do směrovací

tabulky cesta s nejnižší AD. EIGRP má pro interní cesty výchozí hodnotu 90, pro externí 170 a pro sumarizované hodnotu 5 [24].

Pro další výklad je potřebné znát význam následujících pojmů:

- **Successor (S)** - sousední směrovač, přes kterého vede cesta s nejnižší metrikou, splňující Feasibility Condition (viz dále);
- **Feasible Successor (FS)** - sousední směrovač na cestě do cíle, přes něhož vede cesta vyhovující podmínce Feasibility Condition;
- **Feasible Distance (FD)** - dosud nejnižší metrika od směrovače do cíle;
- **Reported Distance (RD)** - metrika do cíle získaná od souseda;
- **Feasibility Condition (FC)** - podmínka používaná při výběru S a FS. Zajišťuje výběr cest beze smyček;
- **Aktivní a pasivní stav cesty** - pasivní stav znamená, že je znám S, síť je zkonvergovaná, cesta je funkční a lze ji použít. Do aktivního stavu se cesta dostane tak, že není znám S a ani FS pro danou cestu.

2.5.1 Tabulka topologie

Tabulka topologie [24] obsahuje informace z konfigurace směrovače a ze směrovacích tabulek od všech sousedů. Pro každý nakonfigurovaný L3 protokol má EIGRP proces samostatnou tabulku topologie. Jsou v ní uchovávány následující informace:

- **Stav cesty** - může být pasivní, aktivní, případně určení typu zprávy, která je odesílána a obsahuje danou cestu;
- **Query origin** a **Reply status** - příznaky definující stav cesty;
- **Zdroj cesty** - zda je cesta interní nebo externí;
- **Next Hop** - síťová adresa sousedního směrovače na cestě do cílové sítě;
- **FD, RD a současná metrika cesty**;
- **Šířka pásma, zpoždění, spolehlivost, zatížení, MTU a Hop count**. Mohou zde být i další parametry, pokud je směrovač dokáže poskytnout;
- **Originating router** - je identifikace směrovače, který cestu vytvořil;
- Externí cesty přidávají další informace, jako název zdroje (např. název směrovacího protokolu), metriku cesty v externím systému a další [3];

2.5.2 Stub směrování

Stub směrování [24] je funkce, která omezuje šíření zpráv *Query*. Limituje také možnosti používání stub směrovače jako následníka (Successor) do cílové sítě. Používá se především v Hub-and-Spoke topologii. Stub směrovač v *Hello* zprávách oznamuje sousedům svůj stav. Používá se speciální TLV položka v EIGRP paketu, kde je zaznamenáno, že daný směrovač je Stub (viz kapitolu 2.1.2). Žádní sousedé mu pak nebudou posílat *Query* zprávy. Stub směrovač sousedům může oznamovat cesty pomocí *Update*, závisí to na nastavení. Avšak odesílání cest ve zprávách *Query* omezeno není.

2.6 Diffusing Update Algorithm

Automat DUAL [21] slouží pro výběr nejlepší cesty beze smyček do cílové sítě. Byl vyvinut v SRI International pod vedením J.J. Garcia-Luna-Aceves. Algoritmus pracuje se sítí jako s neorientovaným grafem, kde uzly jsou směrovače a hrany linky, které je propojují. DUAL využívá difúzních výpočtů, Feasibility Condition (FC) a také definuje reakce na všechny možné události, které se mohou vyskytnout během výpočtu.

Difúzní výpočet slouží pro řízení výměny potřebných informací mezi sousedními uzly. Ke své činnosti používá zprávy *Query* a *Reply*. Výpočet začíná tím, že všem sousedním uzlům je odeslána zpráva *Query*. Příjemci ji mohou delegovat dále, nebo odpovědět zprávou *Reply*. Až jsou shromážděny všechny odpovědi, pak dojde k jejich zpracování a výběru nejlepší cesty.

FC je postačující podmínka pro zajištění, že cesta přes zvoleného následníka (Successor) nebude obsahovat smyčku. Existují tři různě formulované FC, a to Distance Increase Condition (DIC), Current Successor Condition (CSC) a Source Node Condition (SNC). Následuje vysvětlení těchto podmínek [19, str. 132], význam použitých symbolů je v tabulce 2.4.

Pokud uzel i musí změnit S do cílového uzlu j v čase $t > t'$, pak:

- pro DIC při snížení vzdálenosti vybere uzel $k \in N_i(t) : D^i_{jk}(t) + l^i_k(t) = Min\{D^i_{jx}(t) + l^i_x(t) | x \in N_i(t)\} \wedge D^i_{jk}(t) + l^i_k(t) < \infty$. Při zvýšení vzdálenosti musí ponechat nynějšího S;
- u CSC vezme jako S uzel $k \in N_i(t) : D^i_{jk}(t) + l^i_k(t) = Min\{D^i_{jx}(t) + l^i_x(t) | x \in N_i(t)\} \wedge D^i_{jk}(t) \leq FD^i_j(t)$, kde $FD^i_j(t) = D^{*i}_{jv}(t), v = S(t')$. Pokud žádný soused nesplňuje podmínku, pak ponechá současného S;
- při použití SNC zvolí $k \in N_i(t) : D^i_{jk}(t) + l^i_k(t) = Min\{D^i_{jx}(t) + l^i_x(t) | x \in N_i(t)\} \wedge D^i_{jk}(t) < FD^i_j(t)$, kde $FD^i_j(t) = D^{*i}_j(t)$. Pokud takový uzel není nalezen, pak zachová stavající S.

Symbol	Popis
N_i	Množina uzlů, jenž jsou sousedy uzlu i.
S	Successor uzlu i na cestě k cíli j.
l^i_k	Vzdálenost z uzlu i do sousedního uzlu k, jak ji zná i.
D^i_j	Nejkratší vzdálenost z uzlu i do cíle j, jak ji zná i.
D^i_{jk}	Vzdálenost z uzlu k do cíle j, jak ji zná i.
D	Aktuální vzdálenost od uzlu i k cíli j přes uzel S.
D^{*i}_j	Nejmenší dosud známá hodnota D^i_j.
D^{*i}_{jk}	Nejnižší hodnota D^k_j známá uzlu i.
FD^i_j	Vzdálenost, kterou uzel i používá pro ověření FC.
RD^i_j	Vzdálenost z uzlu i do cíle j určená sousedům uzlu i.
origin	Query Origin příznak definující stav cesty.
r^i_{jk}	Reply Status příznak, $r^i_{jk} = 1$ znamená, že uzel i čeká na *Reply* od uzlu k pro cestu do uzlu j. Pokud uzel nečeká na *Reply*, pak je $r^i_{jk} = 0$.

Tabulka 2.4: Symboly pro popis algoritmu

EIGRP používá podmínku Source Node Condition (SNC), jejíž neformální znění je následující. Pokud uzel musí změnit následníka pro daný cíl, pak vybírá mezi sousedními uzly. Zvolí ten uzel, přes něhož vede cesta s nejnižší metrikou. Současně musí platit, že RD od vybraného souseda je menší než FD. Pokud takový uzel není nalezen, pak musí zachovat stávajícího následníka.

2.6.1 Popis algoritmu

Automat je znázorněn na obr. 2.5, k přechodům mezi stavy je připojen doprovodný popis. Pro snazší vysvětlení algoritmu jsou zavedeny symboly, viz tabulku 2.4. Každý přechod je označen číslem a jsou u něj zapsány události, které jej mohou spustit. Za výčtem událostí se nachází v hranatých závorkách podmínka, která musí být při při výskytu události splněna. Automat je popsán z pohledu uzlu i, který zpracovává změny související s cílovým uzlem j. Čerpal jsem především z uvedených zdrojů [19, 21, 18] a vlastních experimentů.

Stav automatu je určen příznaky Query Origin a Reply Status. Query Origin udává stav uzlu i a také to, jak se do něj dostal. Reply Status umožňuje uzlu určit, kterým sousedům odeslal zprávu *Query* a dosud od nich nepřijal *Reply*.

Inicializace uzlu se skládá z nastavení pasivního stavu a vzdálenosti ke všem sousedům na ∞. Vzdálenost k sobě samému nastaví na 0. Uzel v pasivním stavu při výskytu události ověří nejdříve FC. Zkontroluje tedy, zda existuje FS vyhovující této podmínce. Při splnění podmínky proběhne pouze lokální výpočet bez účasti sousedů. Jinak přejde uzel pro danou cestu do aktivního stavu a zahájí difúzní výpočet.

Při objevení nové cesty dojde k její inicializaci, která zahrnuje nastavení vzdálenosti, FD a RD na ∞ a pasivního stavu.

Byla-li ve směrovací tabulce nahrazena EIGRP cesta jinou cestou, pak je tato událost předána ke zpracování. Výsledkem je, že se u cesty změní FD_j^i na 0 (vynutí ztrátu všech cest do cíle) a událost se zpracuje jako změna vzdálenosti. Tento poznatek byl získán z experimentů, jejichž výstupy jsou v příloze A.

Následuje popis přechodů automatu.

1. Uzel zůstává v pasivním stavu pro danou cestu.

   Podmínky přechodu - *Query* od uzlu jiného než S, FC splněna nebo je cesta do cíle neznámá.

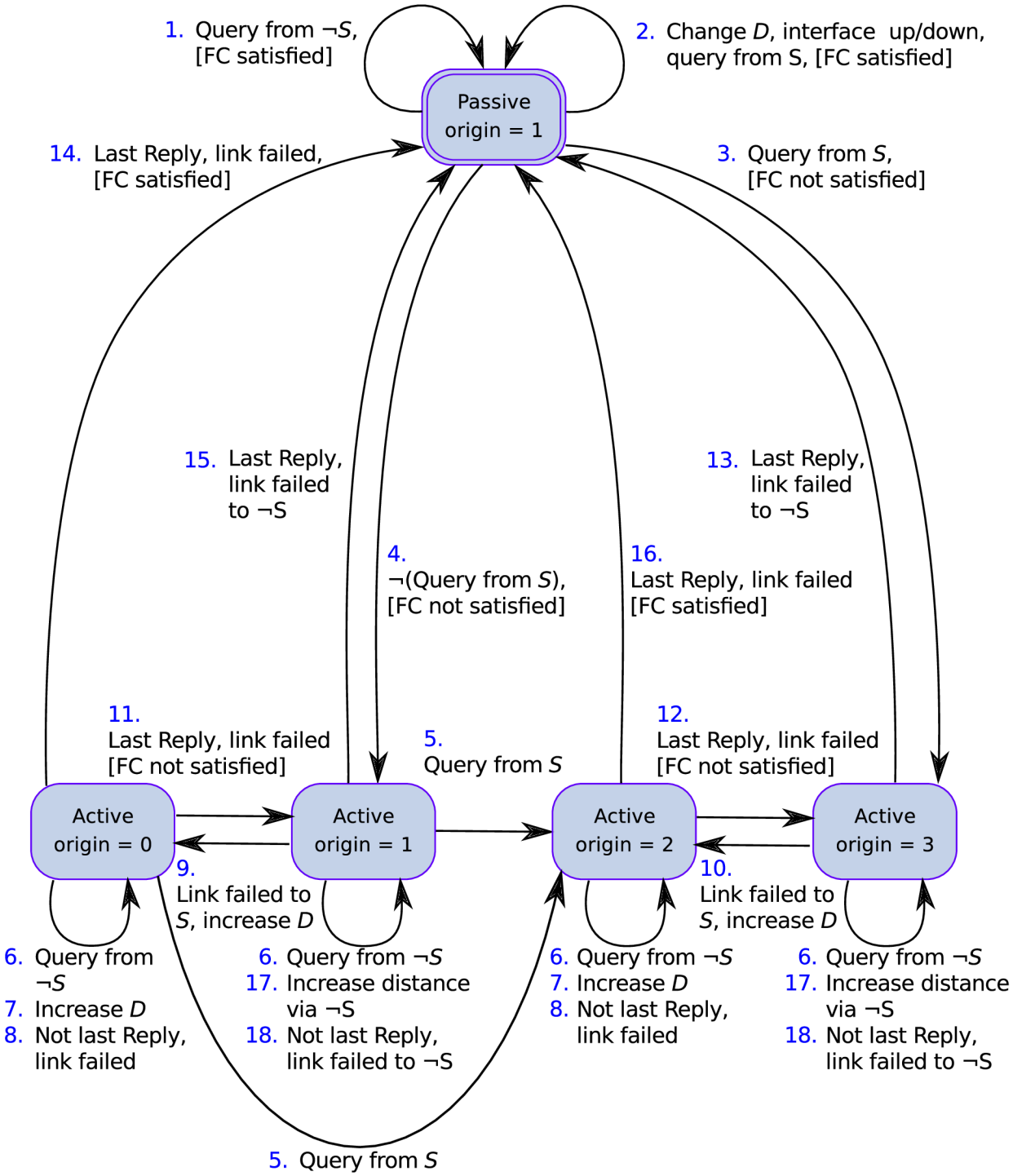
   Akce - uloží informace z *Query* a odešle *Reply* s aktuální vzdáleností RD_j^i. V případě změny vzdálenosti odeslán *Update* sousedům.

2. Cesta setrvá v pasivním stavu.

   Podmínky přechodu - změna vzdálenosti D v uzlu nebo příchozí *Update* s touto změnou, změna stavu přímo připojené linky nebo *Query* od S. FC je splněna.

   Akce - je-li vstupní událost přerušení linky se sousedem k, nastaví aktuální vzdálenost do uzlu k na ∞ ($l_k^i = \infty$ a $D_{jk}^i = \infty$). Pak dojde na lokální výpočet:

   - nalezne mezi všemi cestami tu s nejmenší vzdáleností (*Dmin*) do cíle j;
   - aktualizuje svoji vzdálenost D_j^i a RD_j^i na hodnotu *Dmin*;
   - je-li nová vzdálenost D_j^i menší, než předešlá, pak to bude nová FD_j^i;
   - nastaví S na souseda, přes kterého vede cesta se vzdáleností *Dmin*;
   - pokud je vstupem *Query* od S, pak odešle *Reply* s aktuální hodnotou RD_j^i.



Obrázek 2.5: Diagram automatu DUAL

Při změně D_j^i odeslán *Update* všem sousedům.

3. Přechod do aktivního stavu.

   Podmínky přechodu - přijme *Query* od S, FC není splněna.

   Akce - vypočte vzdálenost k cíli j přes S a uloží ji do FD_j^i a do RD_j^i. Všem sousedům pošle *Query* s vypočítanou vzdáleností.

4. Stav uzlu pro danou cestu je změněn na aktivní.

   Podmínky přechodu - jakákoli událost kromě *Query* odeslaného od S. Tzn., že nastane změna vzdálenosti D v uzlu i nebo přijde *Update* s touto změnou, ztráta spojení s uzlem S či *Query* od kohokoli kromě S. FC není splněna.

   Akce - pokud je vstupem pád linky k uzlu S, pak nastaví vzdálenost k tomuto uzlu na ∞ ($l_s^i = \infty$ a $D_{js}^i = \infty$). Následně provede tyto akce:

   - vypočte vzdálenost k cíli j přes S a její hodnotou aktualizuje D_j^i a RD_j^i;
   - nastaví $FD_j^i = D_j^i(t)$;
   - je-li vstupem příchod *Query*, pak uzel i okamžitě odpoví zprávou *Reply* s RD_j^i;
   - odešle *Query* všem sousedům;

5. Příznak origin změněn na 2.

   Podmínky přechodu - *Query* od uzlu S.

   Akce - žádné akce se neprovádí. Přechod způsobí, že v budoucnu bude třeba odpovědět zprávou *Reply* (viz přechod č. 13 a 16).

6. Uzel zůstává ve stejném stavu.

   Podmínky přechodu - *Query* od souseda, jenž není S.

   Akce - hodnota vzdálenosti z *Query* je zaznamenána a je odeslán *Reply* s $RD_j^i(t)$ (hodnota při přechodu cesty do aktivního stavu).

7. Nadále přetrvává aktivní stav a také hodnota origin příznaku.

   Podmínky přechodu - změna vzdálenosti do cíle j nebo zpráva *Update* se stejnou informací.

   Akce - pouze se aktualizuje hodnota D_j^i.

8. Uzel setrvá v nynějším stavu.

   Podmínky přechodu - ztráta linky k sousedovi k, nebo příchod *Reply* od souseda k. Současně musí platit, že se čeká na odpověď od alespoň jednoho dalšího souseda.

   Akce - pokud byla přerušena linka, pak nastaví vzdálenost přes tohoto souseda na ∞ ($D_{jk}^i = \infty$ a $l_k^i = \infty$). Zaznamená, že nepožaduje od souseda *Reply* ($r_{jk}^i = 0$).

9. Změna příznaku origin na 0.

   Podmínky přechodu - zvýšení vzdálenosti k cíli j, nebo ztráta linky k S.

   Akce - pokud je vstupem ztráta spojení, pak nastaví vzdálenost na ∞ ($D_{jS}^i = \infty$ a $l_S^i = \infty$). V každém případě zaznamá, že nepožaduje *Reply* od S ($r_{jS}^i = 0$). Do D_j^i uloží novou vzdálenost přes S.

10. Nastavení origin na 2.

    Podmínky přechodu - zaznamenáno zvýšení vzdálenosti D, případně zráta spojení s uzlem S.

    Akce - stejné akce, jako při přechodu č. 9.

11. Příznak origin změněn na 1, aktivní stav zůstává.

    Podmínky přechodu - přijat poslední *Reply*, nebo byla přerušena linka k sousedovi, který jako poslední neodeslal *Reply*. FC není splněna s $FD_j^i(t)$ (hodnota při přechodu do aktivního stavu).

    Akce - odešle *Query* všem sousedům.

12. Změna příznaku origin na 3.

    Podmínky přechodu - příjem posledního *Reply*. FC není splněna s $FD_j^i(t)$.

    Akce - odešle *Query* všem sousedům.

13. Uzel přejde do pasivního stavu.

    Podmínky přechodu - přijat poslední *Reply*, nebo pád linky k uzlu, který není S, a současně je poslední, kdo neposlal *Reply*.

    Akce - nastaví FD_j^i na hodnotu ∞. Určí cestu s nejnižší vzdáleností a soused, přes kterého vede tato cesta, se stane S. Odešle bývalému S *Reply* s aktuální hodnotou RD_j^i. Pokud nastala změna D_j^i, pak odešle *Update* všem s touto změnou.

14. Přechod do pasivního stavu.

    Podmínky přechodu - uzel i přijal poslední *Reply*, nebo byla přerušena linka k sousedovi, který jako poslední neodeslal *Reply*. FC je splněna s $FD_j^i(t)$.

    Akce - vypočte nejkratší vzdálenost do cíle j a určí S. Pokud nastala změna D_j^i, pak odešle *Update* všem.

15. Automat přejde do pasivního stavu.

    Podmínky přechodu - poslední *Reply*.

    Akce - nastaví FD_j^i na ∞. Následně se určí S spolu s nejkratší vzdáleností do cíle j. Pokud nastala změna D_j^i, pak odešle *Update* všem o této změně.

16. Aktivní stav změněn na pasivní.

    Podmínky přechodu - poslední *Reply*, FC splněna s $FD_j^i(t)$.

    Akce - nalezne cestu do cíle j s nejkratší vzdáleností a určí S. Bývalému S pošle *Reply* s novou hodnotou RD_j^i, pokud je dostupný. Nastala-li změna D_j^i, pak odešle *Update* všem.

17. Ponechán aktivní stav a také hodnota origin příznaku.

    Podmínky přechodu - změna vzdálenosti do cíle j přes uzel, který není S. Případně zpráva *Update* se stejnou informací.

    Akce - pouze se aktualizuje hodnota D_j^i. Tzn., že se provedou stejné kroky jako při přechodu č. 7.

18. Uzel setrvá v nynějším stavu.

    Podmínky přechodu - příchod *Reply* od souseda k (může být S), nebo pád linky k sousedovi k, který není S. Současně musí platit, že se čeká na odpověď od alespoň jednoho dalšího souseda.

    Akce - pokud byla přerušena linka, pak nastaví vzdálenost přes tohoto souseda na ∞ ($D^i_{jk} = \infty$ a $l^i_k = \infty$). Zaznamená, že nepožaduje od souseda *Reply* ($r^i_{jk} = 0$). Tzn., že se provedou stejné kroky jako při přechodu č. 8.

2.6.2 Pravidla pro omezení smyček

EIGRP kombinuje Split Horizon a Poison Reverse [24] jako prevenci proti smyčkám.

Split Horizon zajišťuje, že se nikdy neodešle zpráva s cestou na rozhraní, které směrovač používá pro dosažení cíle cesty. Pravidlo je aplikováno na zprávy *Update* a *Query*. Nesmí být použito na zprávu *Reply*, jelikož by byl narušen difúzní výpočet.

Poison Reverse je pravidlo, které říká, že směrovač oznámí nedostupnost cesty na rozhraní, které používá pro dosažení cíle cesty. Důsledkem je, že směrovač, který přijme zprávu o nedostupnosti cesty, ji odebere z tabulky topologie. Používá se při odesílání zpráv typu *Update*, *Query* a *Reply*.

2.7 Reliable Transport Protocol

Algoritmus DUAL předpokládá, že přenos dat je bezchybný. Spolehlivý přenos zajišťuje RTP [21], a to pro unicastové i multicastové zprávy nesoucí směrovací informace (*Update*, *Query*, *Reply*). Slouží ale i pro nespolehlivý přenost dat. Používá Automatic Repeat Request metodu Stop and Wait a umožňuje detekovat duplicitní zprávy.

RTP pracuje se sekvenčním číslem (Sequence Number) a číslem potvrzení (Ack Number), které jsou součástí hlavičky EIGRP zprávy. Sekvenční číslo se udržuje pro celý EIGRP proces [25], číslo potvrzení pro každého souseda zvlášť.

Nespolehlivě odeslané zprávy mají sekvenční číslo rovno hodnotě 0. Má-li být zpráva přenesena spolehlivě, pak musí být toto číslo nastaveno na nenulovou hodnotou. Pro potvrzení příjmu spolehlivě přenášené zprávy slouží zpráva *Ack*, která obsahuje sekvenční číslo poslední zprávy přijaté od souseda. Zprávy *Update*, *Query* a *Reply* mohou mít také význam potvrzení stejně jako zpráva *Ack*.

Až směrovač přijme potvrzení na spolehlivě odeslanou zprávu, pak může odeslat další. Pokud potvrzení nepřijde a vyprší časovač RTO, dojde k přeposlání zprávy z fronty zpráv. Tato fronta udržuje zprávy pro opětovné odeslání a je udržována pro každého souseda zvlášť.

Retry limit definuje maximální počet přeposlání zprávy. Pokud došlo k šestnácti přeposlání zprávy a nebylo přijato potvrzení, pak je sousedství se směrovačem ukončeno. Na rychlých linkách se tedy nemusí čekat na vypršení časovače *Hold* [2].

Pacing Packets je omezení rychlosti odesílání EIGRP zpráv tak, aby nedošlo k zahlcení linky. Ve výchozím nastavení se použije nejvýše 50% ze šířky pásma nakonfigurované na rozhraní směrovače.

Conditionally Received (CR) mód [21, kap. 5.2] slouží pro neblokující přenos spolehlivých multicastových zpráv na rozhraních, kde je více než jeden soused. Pro přepnutí směrovače do tohoto módu slouží TLV položka Sequence Type ve zprávě *Hello*, která obsahuje adresy sousedních směrovačů [21, kap. A.7.3]. Příjemce *Hello* ověří, zda je uvnitř jeho adresa. Pokud není, přepne se příjemce do CR módu. Pouze směrovač v CR módu smí zpracovat následně přijatou zprávu s příznakem CR, který je umístěn v hlavičce EIGRP zprávy, viz kapitolu 2.1.1 a tabulku 2.1. Ostatní takovou zprávu musí zahodit a nesmí odeslat potvrzení o přijetí.

2.8 Vyvažování zátěže

Vyvažování zátěže [23] je funkce pro rozložení provozu na více cest do stejné cílové sítě. V Cisco implementaci je tato funkce implicitně zapnuta a nastavený maximální počet cest je obvykle čtyři. Vyvažování lze vypnout nastavením počtu cest na hodnotu 1.

EIGRP dovoluje dva typy vyvažování:

- stejnoměrné (Equal-cost), kdy se do směrovací tabulky vkládájí cesty do stejného cíle, které mají minimální metriku;
- nestejnoměrné (Unequal-cost) je vyvažování zátěže mezi cestami, které nemají stejnou metriku. Uvede se v činnost nastavením parametru *Variance* na hodnotu větší než 1. Pracuje se zde s nejkratší vzdáleností do cíle přes některého ze sousedů (*MinDistance*). Pro vložení cesty, která má vzdálenost do cíle *RouteDistance*, do směrovací tabulky musí platit, že splňuje FC, a také:

$$RouteDistance \leq MinDistance * Variance.$$

Kapitola 3

Konfigurace EIGRP na Cisco zařízeních

Tato kapitola se zabývá konfigurací protokolu EIGRP pro IPv4 a IPv6 na směrovačích od firmy Cisco. Jako reference byla použita příručka pro konfiguraci [8] a další zdroje [24].

Existují dva způsoby konfigurace EIGRP, a to tzv. konfigurace autonomního systému (Autonomous System Configuration) a novější pojmenovaná konfigurace (Named Configuration) [6]. Oba dovolují konfiguraci všech vlastností EIGRP. Chování EIGRP je nezávislé na způsobu konfigurace až na několik výjimek. Je-li EIGRP nastaveno pomocí pojmenované konfigurace, pak směrovač používá rozšířenou metriku. Jinak použije klasickou [11]. Pojmenovaná konfigurace je dostupná od verze EIGRP 8.0 a přenáší všechnu konfiguraci související s EIGRP do tohoto konfiguračního módu. Např. konfigurace související s rozhraními je přemístěna sem. Nejdříve je popsána konfigurace autonomního systému, následně pak pojmenovaná konfigurace.

3.1 Konfigurace pro IPv4

Zde je popsána konfigurace autonomního systému pro IPv4. Základní konfigurace zahrnuje povolení EIGRP a přiřazení čísla procesu (AS). Příkazem `network` a prefixem IP adresy lze určit rozhraní přímo připojené ke směrovači, které chceme zahrnout do EIGRP, a šířit jejich IP adresy v síti. Wildcard mask je invertovaná maska sítě a určuje délku IP prefixu. Je-li síť přidána bez její specifikace, pak se EIGRP chová jako classful protokol. Rozhraní směrovače lze označit jako pasivní, pak nebudou na daném rozhraní odesílány ani přijímány žádné EIGRP zprávy. Lze také zadat Router ID v syntaxi IPv4 adresy. Není povinné, směrovač jej dokáže vybrat sám podle běžných pravidel používaných např. i v OSPF [10].

```
Router(config)# router eigrp as-number
Router(config-router)# network ip-address [wildcard-mask]
Router(config-router)# passive-interface interface
Router(config-router)# eigrp router-id router-id
```

Sousedství mezi směrovači lze nakonfigurovat staticky. Součástí příkazu je IP adresa souseda, který je na daném rozhraní.

```
Router(config-router)# neighbor ip-address outgoing-interface
```

Výpočet metriky lze ovlivnit upravením K-hodnot. Dále jsou uvedeny příkazy pro upravení parametrů *Bandwidth* a *Delay*, jenž se použijí ve výchozím nastavení pro výpočet metriky. Další příkaz slouží pro změnu intervalu, s jakým se sbírají statistiky rozhraní pro určení spolehlivosti a zatížení, které mohou být použity pro výpočet metriky [7]. Posledním příkazem lze ovlivnit maximální šířku pásma rozhraní, kterou protokol pro své zprávy využije.

```
Router(config-router)# metric weights tos K1 K2 K3 K4 K5
Router(config-if)# bandwidth bw
Router(config-if)# delay dly
Router(config-if)# load-interval [counter {1|2|3}] seconds
Router(config-if)# ip bandwidth-percent eigrp as-number percent
```

Následují příkazy pro úpravu časovačů pro udržování sousedství, a to *Hello* a *Hold* časovače. Je doporučeno, aby *Hold* byl alespoň trojnásobkem *Hello* intervalu.

```
Router(config-if)# ip hello-interval eigrp as-number seconds
Router(config-if)# ip hold-time eigrp as-number seconds
```

Dalším příkazem lze vypnout pravidlo Split Horizon, které je ve výchozím stavu zapnuté. Pokud je toto pravidlo vypnuté, pak se neuplatní ani Poison Reverse.

```
Router(config-if)# no ip split-horizon eigrp as-number
```

Automatickou sumarizaci lze vypnout, a je to také doporučeno. EIGRP dovoluje manuální sumarizaci, která se nastavuje pro každé rozhraní samostatně. Hlavním parametrem je sumarizovaná IP adresa s maskou. Pomocí *leak-map* lze specifikovat cesty, jenž budou oznamovány samostatně. Tzn. nebudou zahrnuty do sumarizace.

```
Router(config-router)# no auto-summary
Router(config-if)# ip summary-address eigrp as-number ip-address mask
   [admin-distance] [leak-map name]
```

EIGRP poskytuje funkce pro vyvažování zátěže. Lze definovat maximální počet cest, jež budou vloženy do směrovací tabulky. Pro nestejnoměrné vyvažování zátěže je nutné nastavit parametr *variance* na hodnotu multiplikátoru větší než 1.

```
Router(config-router)# maximum-paths number
Router(config-router)# variance multiplier
```

Směrovač je možné označit jako stub a snížit tak zatížení sítě, využívá se často v topologii Hub-and-Spoke. Příkaz bez parametrů má stejný význam jako při zadání parametrů *connected* a *summary*. Popis parametrů je uveden v kapitole 2.1.2 v tabulce 2.2.

```
Router(config-router)# eigrp stub [receive-only|connected|static|
   summary|redistributed|leak-map name]
```

3.2 Konfigurace pro IPv6

Konfigurace EIGRP pro IPv6 je velmi podobná té pro IPv4. Z tohoto důvodu jsou zde uvedeny jen příkazy, které se liší od těch pro IPv4. Čerpal jsem z uvedených zdrojů [5].

Před konfigurací směrovacího protokolu je nutné povolit funkci směrování IPv6 provozu. Dále je potřeba vytvořit EIGRP proces a explicitně přiřadit směrovači Router ID (RID) v syntaxi IPv4 adresy, který musí být mezi směrovači v AS jedinečný. Pak je nutné ještě proces spustit.

```
Router(config)# ipv6 unicast-routing
Router(config)# ipv6 router eigrp as-number
Router(config-rtr)# router-id ip-address
Router(config-rtr)# no shutdown
```

Rozhraní se do EIGRP nepřidávají na úrovni směrovacího protokolu, jako u EIGRP pro IPv4, ale přímo na rozhraní.

```
Router(config-if)# ipv6 eigrp as-number
```

3.3 Pojmenovaná konfigurace

Pojmenovaná konfigurace je trochu odlišná od konfigurace autonomního systému. Níže jsou uvedeny základní příkazy [8].

EIGRP proces je vytvořen prvním z příkazů, dalším se přiřadí rodina adres (address-family) a spustí se proces. Ostatní příkazy v address-family módu jsou podobné těm v konfiguraci autonomního systému.

```
Router(config)# router eigrp name
Router(config-router)# address-family {ipv4|ipv6} [multicast] [unicast]
   [vrf vrf-name] autonomous-system as-number
Router(config-router-af)# network ip-address wildcard-mask
Router(config-router-af)# metric weights tos K1 K2 K3 K4 K5 K6
Router(config-router-af)# eigrp stub [receive-only|connected|static|
   summary|redistributed|leak-map name]
```

V módu pro nastavení rozhraní mohou být nakonfigurovány další parametry, opět jsou uvedeny jen ty nejčastěji používané.

```
Router(config-router-af)# af-interface interface
Router(config-router-af-interface)# passive-interface
Router(config-router-af-interface)# bandwidth-percent percent
Router(config-router-af-interface)# hello-interval seconds
Router(config-router-af-interface)# hold-time seconds
Router(config-router-af-interface)# no split-horizon
Router(config-router-af-interface)# summary-address ip-address mask
   [leak-map name]
```

Poslední konfigurační mód je určen pro nastavení základní topologie (parametr base). Případně lze nakonfigurovat logické topologie, které používá funkce Multi-Topology Routing [13].

```
Router(config-router-af)# topology {base|topology-name tid number}
Router(config-router-af-topology)# no auto-summary
Router(config-router-af-topology)# maximum-paths number
Router(config-router-af-topology)# variance multiplier
```

3.4 Ověření konfigurace

Zde uvádním hlavní příkazy pro ověření správnosti konfigurace protokolu EIGRP pro IPv4. U EIGRP pro IPv6 se příkazy odlišují jen tím, že se u nich specifikuje prokol IPv6.

Následujícím příkazem lze vypsat nastavení všech směrovacích protokolů spuštěných na směrovači. Součástí výpisu je číslo autonomního systému, K-hodnoty, informace o sumarizaci, sítě oznamováné směrovacím protokolem, sousední směrovače a další.

```
Router#show ip protocols
```

Zobrazením tabulky sousedství lze vypsat směrovače, se kterými byl ustaven tento vztah. Součástí výpisu je síťová adresa souseda, rozhraní, na kterém je připojen, hodnota časovače *Hold*, poslední sekvenční číslo od souseda a další.

```
Router# show ip eigrp neighbors
IP-EIGRP neighbors for process 1
H   Address         Interface       Hold Uptime   SRTT   RTO  Q   Seq
                                    (sec)         (ms)       Cnt  Num
1   192.168.10.6    Se0/1           10  00:01:09   52   312  0    8
0   172.16.3.2      Se0/0           14  00:01:09   33   200  0    9
```

Výpis tabulky topologie ukazuje stav všech cest. U každé cesty je uveden Successor, Feasible Distance a stav cesty. Cesta může být ve stavu aktvním, pasivním, případně je uveden typ odeslané zprávy. Také je tu číslo autonomního systému (AS) a Router ID směrovače. Pokud je příkaz doplněn o IP adresu a masku cesty, pak jsou zobrazeny parametry cesty pro výpočet metriky, MTU, Hop Count a další.

```
Router# show ip eigrp topology
IP-EIGRP Topology Table for AS(1)/ID(192.168.10.5)

Codes: P - Passive, A - Active, U - Update, Q - Query, R - Reply,
       r - reply Status, s - sia Status

P 192.168.10.4/30, 1 successors, FD is 2169856
        via Connected, Serial0/1
P 192.168.1.0/24, 1 successors, FD is 2172416
        via 192.168.10.6 (2172416/28160), Serial0/1
...
```

Dále je možné zíkat informace o EIGRP rozhraních. Tabulka rozhraní obsahuje především informace spojené s přenosem zpráv. Je zde například počet sousedů na rozhraní, počet zpráv čekajících na odeslání (Xmit Queue), doba čekání před odesláním zprávy (Pacing Time) a další.

```
Router# show ip eigrp interfaces
IP-EIGRP interfaces for process 1

                   Xmit Queue   Mean   Pacing Time   Multicast    Pending
Interface    Peers  Un/Reliable  SRTT   Un/Reliable   Flow Timer   Routes
Se0/0          1        0/0        18       0/15          75           0
Se0/1          1        0/0        25       0/15          79           0
```

Hodnoty pro výpočet metriky lze získat z výpisu informací o rozhraní.

```
Router# show interfaces s0/0
Serial0/0 is up, line protocol is up
  Hardware is M4T
  Internet address is 172.16.3.1/30
  MTU 1500 bytes, BW 1544 Kbit, DLY 20000 usec,
     reliability 255/255, txload 1/255, rxload 1/255
...
```

Pro získání podrodrobnějších informací o činnosti EIGRP protokolu lze povolit ladící výpisy. Také se dá zobrazit výpis všech událostí, které v EIGRP nastaly.

```
Router# debug eigrp {fsm | neighbors | nsf | packets | transmit}
Router# show ip eigrp events
```

Kapitola 4

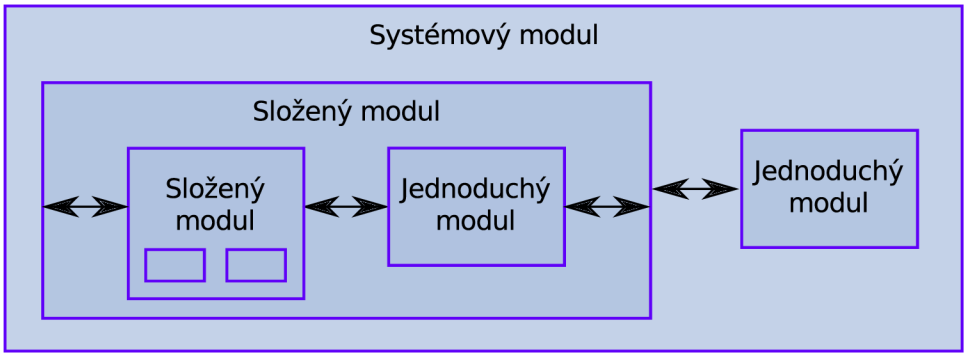
OMNeT++ a knihovna INET

V této kapitole je popsán simulační nástroj OMNeT++ a framework INET. OMNeT++ [26] je nástroj pro diskrétní simulaci sítí. Je napsán v jazyce C++ a je objektově orientovaný. Obsahuje simulační jádro, komponenty připravené pro použití, GUI a další knihovny. Simulátor má modulární strukturu a je tak velmi dobře rozšiřitelný. Ke knihovně existuje řada frameworků, které poskytují specifické vlastnosti pro danou oblast zájmu. Příkladem může být modelování sítě na čipu, internetových protokolů, drátových, bezdrátových a mobilních sítí.

Hierarchická struktura simulačního modelu je znázorněna na obrázku 4.1. Model je složen z modulů, které mohou mít vstupní a výstupní brány pro komunikaci s okolím. Propojení modulů je zajištěno pomocí spojení mezi branami. Spojit lze moduly na stejné úrovni a modul s jeho podmodulem.

Pro popis topologie modelu je používán jazyk NED. Ten slouží pouze pro popis struktury, nikoli chování. Model sítě je složen ze systémového modulu, který je na nejvyšší úrovni, a může obsahovat jednoduché nebo složené moduly. Složené moduly sestávají z dalších složených nebo jednoduchých modulů. Jednoduché moduly nelze dále dělit, jsou to C++ třídy, které definují chování.

V jazyce NED se obvykle definuje také struktura zpráv. OMNeT++ umožňuje z daného popisu vygenerovat C++ třídu.



Obrázek 4.1: Hierarchie modulů v OMNeT++

4.1 INET

INET [17] je framework pro nástroj OMNeT++, jenž jej doplňuje o funkce související s počítačovými sítěmi. Obsahuje implementaci protokolu Ethernet, PPP, IPv4, IPv6, TCP, UDP, OSPFv2 a dalších. Přejímá koncepci OMNeT++ s moduly, které komunikují pomocí předávání zpráv. Moduly jsou kombinovány pro vytvoření síťových zařízení. Součástí frameworku jsou hotové modely základních síťových zařízení a také některých protokolů. Dále jsou zde např. moduly pro uložení dat a konfiguraci.

Projekt ANSA [1] rozšiřuje INET o další protokoly, např. RIP, RIPng, IGMP, PIM a STP. Také jsou zde připraveny modely směrovače, přepínače a počítače.

4.2 Připravenost pro EIGRP

INET a ANSA nabízejí většinu podstatných modulů, které EIGRP potřebuje [16].

Pro včlenění EIGRP do hierarchie modulů v INET jsou důležité především moduly `NetworkLayer` a `NetworkLayer6`, jelikož protokol pracuje nad síťovou vrstvou. Konkrétně se jedná o podmoduly `IPv4` a `IPv6`, které jsou připojené na protokoly na transportní vrstvě. Bude také využita tabulka s informacemi o síťových rozhraních směrovače, která je zastoupena modulem `EigrpInterfaceTable`. Zde chybí parametry rozhraní, jako šířka pásma, zpoždění, spolehlivost a zatížení, které EIGRP používá pro výpočet metriky. Ty bude nutné doplnit do třídy `InterfaceEntry`. Dále bude používána směrovací tabulka v modulu `AnsaRoutingTable`, kde nejspíše nebudou potřeba žádné větší úpravy.

Pro konfiguraci v rámci projektu ANSA vznikl modul `DeviceConfigurator`, který načítá nastavení ve formátu XML. Měl by poskytovat funkce pro automatické generování souboru z reálné konfigurace. Bude využit pro nastavení parametrů EIGRP [22].

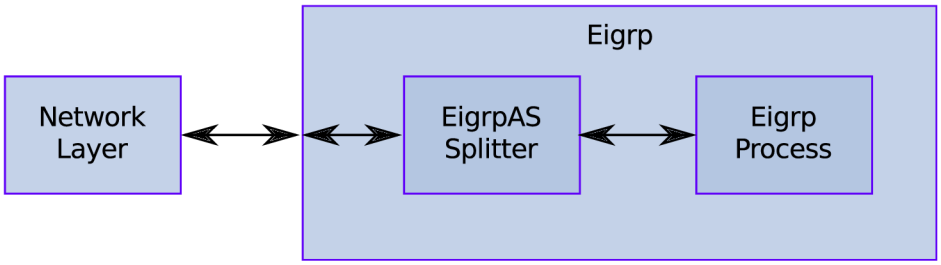
Aby bylo možné simulovat běžné události v síti a základní změny konfigurace protokolu, bude využit modul `InterfaceStateManager`. Ten umožňuje naplánovat zapnutí a vypnutí rozhraní. Sem bude doplněna možnost změny parametrů rozhraní, tj. především šířky pásma a zpoždění.

Kapitola 5

Návrh EIGRP protokolu

Tato kapitola popisuje návrh protokolu EIGRP a jeho napojení na model směrovače. Jako referenční byla zvolena verze EIGRP 10.0, která podporuje všechny důležité vlastnosti tohoto protokolu. Příkladem může být rozšířená metrika nebo stub směrování.

EIGRP pracuje na síťové vrstvě, proto bude napojen na modul `NetworkLayer`. Na směrovači může být nakonfigurováno více instancí EIGRP protokolu, kde každá bude zastoupena modulem `EigrpProcess` a musí mít jedinečné číslo EIGRP autonomního systému (AS). K rozlišení zpráv pro jednotlivé procesy slouží rozbočovač `EigrpAsSplitter`, který podle čísla autonomního systému zprávu odešle správnému procesu. Musí být také možné vytvářet nové EIGRP procesy a připojit je k `EigrpAsSplitter`.



Obrázek 5.1: Napojení EIGRP na model směrovače

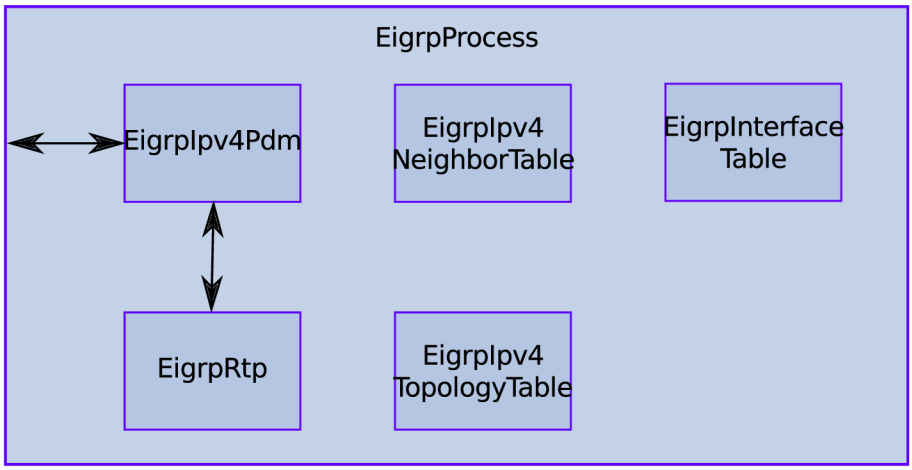
5.1 EIGRP proces

Struktura `EigrpProcess` je naznačena na obrázku 5.2. Protokol byl navržen jako skupina čtyř nezávislých částí. Patří sem `EigrpIpv4Pdm`, který se stará o věci specifické pro protokol IPv4. Mimo jiné vytváří a odesílá EIGRP zprávy do sítě a také je přijímá. Dále je tu `EigrpRtp`, jenž obsahuje funkce pro zajištění spolehlivého i nespolehlivého přenosu zpráv. Tabulka `EigrpIpv4NeighborTable` slouží pro uchování záznamů o sousedních směrovačích, `EigrpIpv4NeighborTable` pro uložení cest a `EigrpInterfaceTable` obsahuje informace o rozhraních, jež jsou povoleny pro EIGRP.

`EigrpIpv4Pdm` přijímá zprávy ze sítě, zpracovává je a přeposílá do `EigrpRtp`. Ten z hlavičky zprávy vybere potřebné informace (např. sekvenční číslo a číslo potvrzení) pro zajištění spolehlivého přenosu zpráv.

Pokud je potřeba odeslat EIGRP zprávu, pak `EigrpIpv4Pdm` vytvoří požadavek a předá

jej do modulu `EigrpRtp`. Ten ho vloží do fronty požadavků. Požadavek obsahuje typ zprávy, identifikaci cíle zprávy a případně seznam identifikátorů odesílaných cest. Až je vhodný čas pro odeslání zprávy, pak `EigrpRtp` odešle požadavek zpět do `EigrpIpv4Pdm`, který z něj vytvoří zprávu a odešle ji do sítě.



Obrázek 5.2: Složený modul EIGRP procesu

5.2 Abstraktní datové typy

Protokol pro svoji činnost vyžaduje tabulku topologie a tabulku sousedů. Navíc byla přidána tabulka rozhraní s informacemi pro EIGRP.

- Tabulka topologie je zastoupena strukturou `EigrpIpv4TopologyTable` a bude uchovávat informace o cestách do cílových sítí, které jsou oznamovány pomocí EIGRP. Položka tabulky zahrnuje informace o stavu cesty, IP adresu Next Hop směrovače, FD, RD, aktuální hodnotu metriky a složky, ze kterých se počítá. Jelikož EIGRP podporuje vyvažování zátěže, může vést do jedné sítě více cest označených jako Successor. Zde je nutné při oznamování směrovacích informací sousedům vybrat jednu cestu:

  1. cesta musí být označena jako Successor;
  2. pokud je takových cest více, pak vybere tu s nejnižší metrikou. To se může stát, je-li nakonfigurováno nestejnoměrné vyvažování zátěže;
  3. posledním kritériem je IP adresa v položce Next Hop cesty. Vybere se cesta s nejnižší IP adresou.

  Toto chování bylo pozorováno na Cisco směrovačích, výstupy jsou uvedené v A.

- `EigrpIpv4NeighborTable` je struktura s informacemi o směrovačích se spuštěným protokolem EIGRP, se kterými byl navázán vztah sousedství. Bude obsahovat označení rozhraní a IP adresu souseda, jeho stavu (up nebo pending), časovač *Hold* pro udržování vztahu sousedství, SRTT a RTO a poslední sekvenční číslo přijaté od sousedního směrovače. Také bude obsahovat stub konfiguraci souseda.

- Na reálném směrovači lze na každém rozhraní nakonfigurovat parametry vztahující se ke kterémukoli EIGRP procesu. Proto byla přidána tabulka s rozhraními přímo do EIGRP, a tedy se nevyužila stávající globální tabulka rozhraní na směrovači. `EigrpInterfaceTable` obsahuje informace o EIGRP rozhraních. Položka tabulky obsahuje identifikaci rozhraní, nakonfigurovaný interval pro časovač *Hold* a *Hello*, dále nastavený *Hello* časovač, nastavení pravidla Split Horizon a to, zda je rozhraní pasivní.

5.3 Návrh PDM pro IPv4

Specifické věci pro protokol IPv4 jsou umístěny do PDM (Protocol Dependent Module). `EigrpIpv4Pdm` bude provádět ustavení a udržování vztahu sousedství, vytváření zpráv, výpočet metriky a upozornění automatu DUAL na události související se směrováním. Zde by měla být také uplatněna pravidla Split Horizon a Poison Reverse, jelikož tento modul bude mít k dispozici potřebné informace o sousedech a rozhraních. DUAL by měl přes tento modul přistupovat k tabulce topologie a ke směrovací tabulce.

Činnost po zapnutí je v diagramu 5.3. Inicializace spočívá v načtení konfigurace a také přihlášení multicastové adresy 224.0.0.10 pro odběr EIGRP zpráv na povolených rozhraních. Rovněž je v diagramu znázorněn proces ustavení a udržování sousedství.

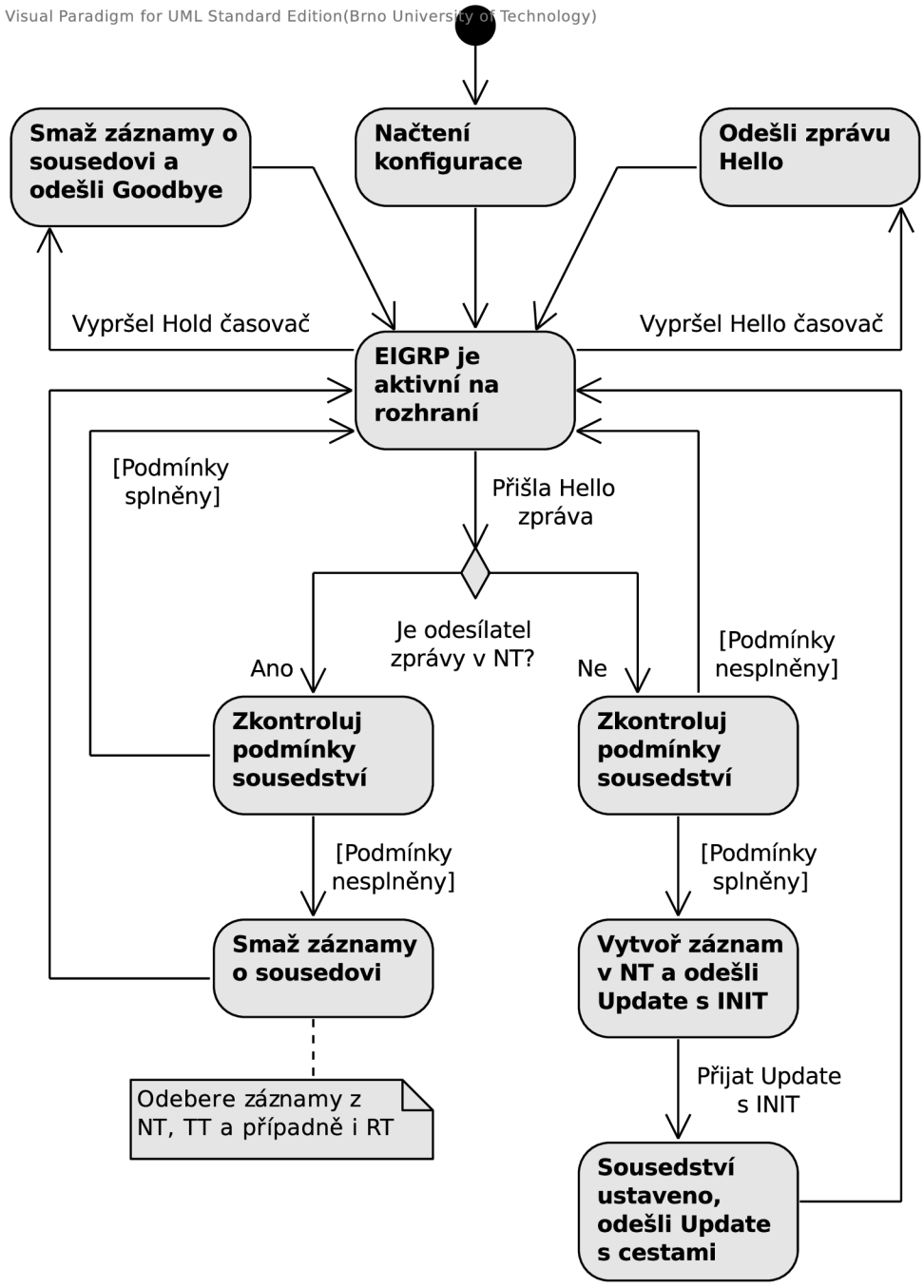
5.3.1 Zpracování událostí

Na směrovači může dojít k událostem, o kterých musí být protokol EIGRP informován. Jedná se především o změny ve směrovací tabulce, změny stavu a parametrů rozhraní, nebo přijetí EIGRP zpráv. Všechny tyto událostí přijímá modul `EigrpIpv4Pdm` a případně je doručí automatu DUAL. Informace předávané automatu obsahují typ (*Query*, *Update*, *Reply*, změna metriky) a cestu (adresa sítě, Next Hop, stav cesty) s nově vypočítanou metrikou.

Při příchodu zprávy *Update*, *Query* nebo *Reply* budou postupně všechny cesty ze zprávy předány automatu DUAL ke zpracování. Ovšem některé cesty ze zprávy *Update* nejsou v určitých případech předány automatu, ale jsou zahozeny. Takto může být ovlivněn obsah tabulky topologie a také chování EIGRP při nějaké události. Toto chování bylo zjištěno na základě provedených testů, jejichž výstupy lze nalézt v příloze A. Situace, kdy se cesta z příchozí zprávy *Update* zahodí, jsou následující:

- cesta není nalezena v tabulce topologie a hodnota Router ID (RID) v přijaté cestě se shoduje s RID směrovače;
- cesta není nalezena v tabulce topologie a současně je přijatá cesta nedostupná;
- cesta je nalezena v tabulce topologie, ale nepřináší žádnou změnu. Tedy metrika cesty ve zprávě je shodná s metrikou cesty v tabulce topologie.

Při vypnutí rozhraní je spuštěna procedura pro odebrání rozhraní z EIGRP. Procedura je blíže popsána na konkrétních situacích v příloze A, z těchto výstupů byla také vypozorována. Nejdříve je předána přímo připojená cesta na tomto rozhraní automatu DUAL, kterou tímto směrovač ztratil. Následně jsou odebráni všichni sousedé na tomto rozhraní. Každá cesta, která vede přes některého z těchto sousedů, je předaná automatu DUAL s informací o nedostupnosti. Pokud DUAL čeká na *Reply* od odebíraného souseda, pak se musí zaznamenat, že odpověď nepřijde.



Obrázek 5.3: Návrh chování modulu EigrpIpv4Pdm po spuštění

Je-li rozhraní naopak zapnuto, pak je postup následující. Nejdříve jsou provedeny změny ve směrovací tabulce bez účasti EIGRP. Přímo připojená cesta z rozhraní je tedy vložena do směrovací tabulky. Zde může nastat případ, že v tabulce již existuje cesta do dané sítě. V tomto případě musí být cesta nejdříve smazána. Pokud smazaná cesta pochází z EIGRP, pak je tento protokol upozorněn na její ztrátu, a událost je předána automatu DUAL. Následuje upozornění EIGRP na nové rozhraní, mechanismu DUAL je předána

nová přímo připojená cesta. Pokud není rozhraní pasivní, pak je nastaven časovač *Hello*, a po jeho vypršení může začít odesílání zpráv *Hello* na toto rozhraní. Tento postup pro zapnutí rozhraní je blíže popsán v A, kde jsou také výstupy Cisco zařízení.

Nastane-li na rozhraní změna některého z parametrů, který se používá pro výpočet metriky, pak se projdou všechny cesty v tabulce topologie, jenž vedou přes toto rozhraní. Cesty jsou předávány s nově vypočítanou metrikou automatu DUAL, jenž řídí další postup.

5.3.2 Uplatnění pravidel pro omezení smyček

`EigrpIpv4Pdm` bude při odesílání zpráv aplikovat pravidla Split Horizon a Poison Reverse. Následuje popis situací, kdy jsou pravidla uplatněna. Všechny zde uvedené případy použití pravidla Split Horizon a Poison Reverse byly vypozorovány z odchycené komunikace a ladících výpisů. Výstupy, ze kterých jsem čerpal, lze nalézt v příloze A.

Split Horizon se použije v následujících případech:

- při odesílání zprávy *Query*, pokud přechod do aktivního stavu způsobila zpráva *Query* přijatá od souseda, jenž je Successor;
- při odesílání *Update* se všemi cestami ze směrovací tabulky, tj. jako odpověď na zprávu *Update* s příznakem INIT;
- pokud nastala změna metriky cesty ve směrovací tabulce (cesta je označena jako Successor), která není přímo připojená (Connected), a cesta má být odeslána ve zprávě *Update*. Pokud současně se změnou metriky došlo ke smazání nebo přidání této cesty do směrovací tabulky, pak se Split Horizon nepoužije.

Pravidlo Poison Reverse se uplatní při odeslání:

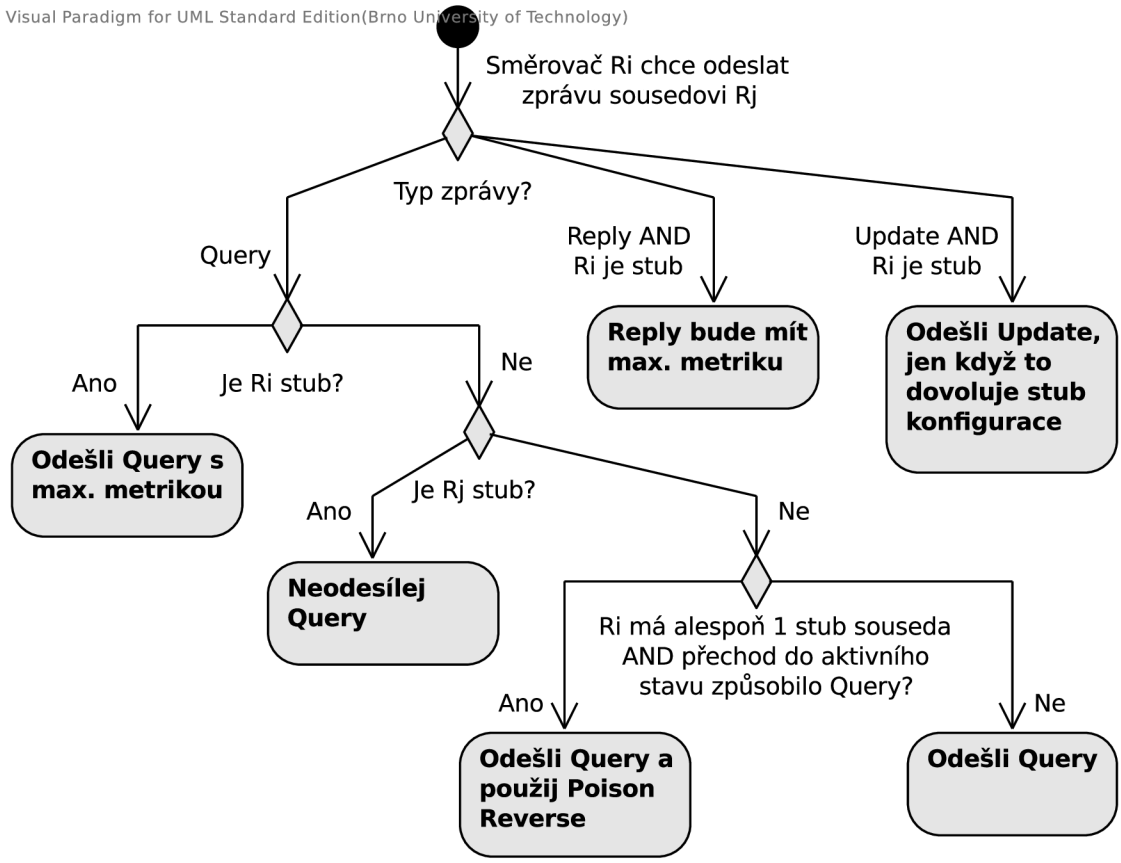
- *Reply* s cestou, která má jako Next Hop souseda, jenž je Successor. Současně musí platit, že je zpráva adresována tomuto sousedovi, který je uveden jako Next Hop;
- použije se u zprávy *Query*, a to pouze pokud přechod do aktivního stavu způsobila zpráva *Update* nebo změna na směrovači (tzn. nesmí to být *Query*);
- je-li odesílána zpráva *Update* s cestou, která byla vložena nebo smazána ve směrovací tabulce. Poison Reverse se neuplatní v případě, že se jedná o cestu, která nemá nejmenší metriku ze všech cest do dané sítě (tj. při použití nerovnoměrného vyvažování zátěže).

5.3.3 Stub směrování

Funkce stub směrování byla popsána v kapitole 2.5.2. Stub směrování ovlivní chování jak směrovačů označených jako stub, tak i těch obyčejných. V diagramu 5.4 je popsáno chování při odesílání EIGRP zpráv. Bližší popis tohoto chování a provedené testy lze nalézt v příloze A.

Zde je dobré si všimnout toho, že stub směrovač může odesílat *Query* jinému stub směrovači. Ovšem cesty v *Query* budou vždy nedostupné. Když stub směrovač přijme *Query*, pak vždy odpoví zprávou *Reply* s informací o nedostupnosti cesty.

Pokud směrovač, který není stub, ztratí cestu do cílové sítě, pak přejde do aktivního stavu a odešle sousedům *Query*. Neodešle tuto zprávu ovšem stub sousedům. Aby se i tyto směrovače dozvěděly o ztrátě cesty, musí směrovač, jenž není stub, při přechodu z aktivního stavu do pasivního stavu odeslat *Update* s informací o nedostupnosti cesty všem stub směrovačům. Pravidlo se použije jen pokud cesta do cíle není nalezena.



Obrázek 5.4: Návrh funkce stub

5.3.4 Správa cest

Tabulku topologie spravuje `EigrpIpv4Pdm`. DUAL rozhoduje o tom, kdy budou cesty v tabulkách upraveny, ale samotné úpravy bude provádět `EigrpIpv4Pdm`. V tabulce topologie mohou být smazány pouze cesty, které mají metriku nastavenou na maximum. Ovšem někdy cesta nesmí být smazána. Taková situace nastane, když je potřeba odeslat zprávu *Reply* s cestou, která má maximální metriku. Cesta může být smazána až po přijetí potvrzení od souseda pro zprávu *Reply*.

Správu směrovací tabulky také provádí tento modul. Na základě informací o cestách v tabulce topologie určuje, které z nich budou vloženy do směrovací tabulky. Při vkládání se musí brát ohled na cesty z jiných zdrojů s nižší administrativní vzdáleností.

5.4 Návrh RTP

Spolehlivý přenos má na starosti modul `EigrpRtp`, který udržuje sekvenční číslo směrovače společné všem zprávám, frontu zpráv, umožňuje přeposlání zprávy při vypršení RTO a Packet Pacing. Také řídí potvrzování spolehlivě přenášených zpráv. Pro tuto funkci využije zprávu *Ack*, případně spolehlivé zprávy čekající na odeslání ve frontě.

RTP přijímá požadavky na odeslání zpráv od `EigrpIpv4Pdm` a vkládá je do fronty pro zprávy. Až bude zpráva na řadě, pak může být odeslána. Požadavek bude mít pevnou

velikost a bude obsahovat indexy na části zprávy. Například cestu zde bude zastupovat jen její index, nikoli data z ní. Díky tomu budou ve zprávě vždy aktuální informace, jelikož zpráva bude vytvořena těsně před odesláním. Tento postup používá ve své implementaci i Cisco [24].

Má-li být zpráva přenesena spolehlivě, pak musí být sekvenční číslo nastaveno na nenulovou hodnotou. Následně směrovač inkrementuje své sekvenční číslo. Také si poznačí, kteří sousedé mu musí potvrdit příjem. Dokud nejsou přijata všechna potvrzení, nejsou na rozhraní odesílány žádné další zprávy. Pokud potvrzení nepřijde a vyprší časovač RTO, dojde k přeposlání zprávy z fronty pro zprávy. Tato fronta je udržována pro každého souseda zvlášť. Platí, že spolehlivé zprávy čekající ve frontě nesmí blokovat zprávy odesílané nespolehlivým způsobem.

5.5 Návrh konfigurace EIGRP

Konfigurace EIGRP bude ve formátu XML. Při návrhu struktury jsem vycházel ze způsobu, jakým je EIGRP konfigurováno na Cisco zařízeních. Struktura již byla definována v jiných pracích [22], zde budou rozšířeny sekce `Routing` a `Interface`.

5.5.1 Konfigurace na úrovni směrovacího procesu

Tady uvádím značky, které se mohou vyskytovat uvnitř `Routing`. EIGRP konfigurace je umístěna v elementu `EIGRP`. Do něj lze vkládat značky `ProcessIPv4` pro vytvoření a spuštění EIGRP procesů pro protokol IPv4. Každý proces musí mít přiřazeno číslo autonomního systému, které se zadá do parametru `asNumber`.

```
<EIGRP>
   <ProcessIPv4 asNumber='value'>
      ...
   </ProcessIpv4>
<EIGRP>
```

Ve zbývající části této podkapitoly jsou uvedeny značky, které se mohou vyskytovat uvnitř elementu `ProcessIpv4`.

V elementu `Networks` může být libovolně mnoho značek `Network`, které slouží pro určení rozhraní, jenž budou součástí EIGRP. Pomocí `PassiveInterface` lze nastavit pasivní rozhraní.

```
<Networks>
   <Network>
      <IPAddress>ip-address</IPAddress>
      <Wildcard>wildcard</Wildcard>
   </Network>
</Networks>
<PassiveInterface>interface-name</PassiveInterface>
```

Následuje element pro nastavení K-hodnot zadávaných pomocí atributů, kde *value* může nabývat 0 až 255. Příkaz není povinný, ani parametry nemusí být zadány všechny. Výchozí hodnoty jsou `K1` = `K3` = 1, `K2` = `K4` = `K5` = `K6` = `tos` = 0.

```
<MetricWeights k1='value' k2='value' k3='value' k4='value' k5='value'
   k6='value' tos='value' />
```

Element `MaximumPath` určuje počet cest, které mohou být vloženy do směrovací tabulky. Další značka slouží pro nastavení nestejnoměrného vyvažování zátěže.

```
<MaximumPath>vaule</MaximumPath>
<Variance>value</Variance>
```

`Stub` je další nepovinný element, kterým se nastavuje funkce stub směrování. Všechny atributy nemusí být uvedeny, zde je ukázáno výchozí nastavení při zápisu bez atributů.

```
<Stub receiveOnly='disabled' connected='enabled' static='disabled'
   summary='enabled' redistributed='disabled' leakMap='disabled' />
```

5.5.2 Konfigurace na úrovni rozhraní

Tato kapitola popisuje XML elementy související s EIGRP, které se mohou vyskytovat v `Interface`.

EIGRP pro výpočet metriky potřebuje některé parametry rozhraní. Sem patří hlavně šířka pásma a zpoždění. Ty lze nastavit pomocí následujících značek, do nichž se vkládájí hodnoty v jednotkách, které používá Cisco.

```
<Bandwidth>value-in-kbps</Bandwidth>
<Delay>value-in-tens-microsec</Delay>
```

Nastavení pro EIGRP proces je v elementu `EIGRP-IPv4`, kde se specifikuje číslo autonomního systému. Zde lze nastavovat interval pro časovač *Hello* a *Hold*. Také je možné povolit nebo zakázat pravidlo Split Horizon, které je ve výchozím stavu povoleno. S tímto pravidlem je spjato Poison Reverse, takže při zakázaní Split Horizon je vypnuto i Poison Reverse.

```
<EIGRP-IPv4 asNumber='value'>
   <HelloInterval>value-in-seconds</HelloInterval>
   <HoldInterval>value-in-seconds</HoldInterval>
   <SplitHorizon>enabled</SplitHorizon>
</EIGRP-IPv4>
```

Kapitola 6

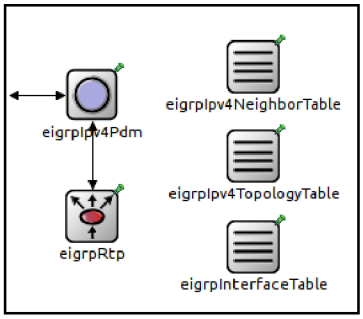
Implementace EIGRP v OMNeT++

V této kapitole je popsána implementace směrovacího protokolu EIGRP pro simulátor OMNeT++ a knihovnu INET. Programovacím jazykem je C++. Diagram tříd, kterým jsem se řídil při implementaci, je v příloze C. Nejdříve budou popsány úpravy v INET mimo EIGRP modul, pak bude uvedena implementace tohoto modulu.

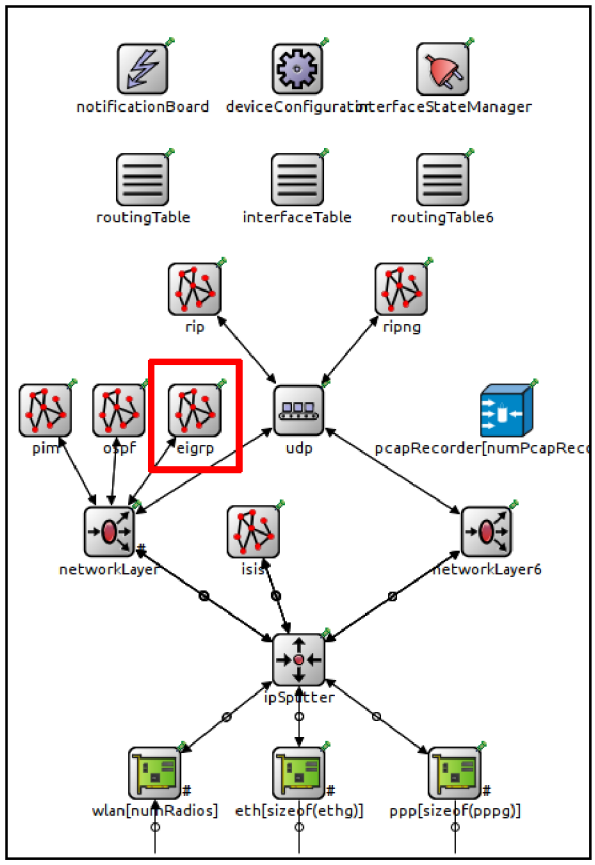
V knihovně INET a ANSAINET byly provedeny drobné úpravy pro potřeby EIGRP modulu. Jedná se o přidání parametrů do třídy `InterfaceEntry`, a to parametru pro šířku pásma, zpoždění, spolehlivost a zatížení, které EIGRP používá pro výpočet metriky. Dále byla rozšířena třída `InterfaceStateManager`, kde jsem přidal možnost naplánovat změnu parametru zpoždění a šířky pásma rozhraní. S tím souvisí i úprava `AnsaRoutingTable`, kde bylo potřeba zajistit správnou reakci na zapnutí a vypnutí rozhraní. Možnost naplánovat změnu parametrů rozhraní při simulaci mi dovolila otestovat implementaci automatu DUAL.

6.1 Modul EIGRP

Na obrázku 6.2 je vidět umístění EIGRP v modelu směrovače `ANSARouter`. EIGRP je napojeno na síťovou vrstvu `AnsaNetworkLayer`. Mimo ní využívá pro svoji činnost směrovací tabulku `AnsaRoutingTable` a tabulku rozhraní `InterfaceTable`. Struktura modulu EIGRP je znázorněna v obr. 6.1. Jak je vidět, jedná se o složený modul.



Obrázek 6.1: Model EIGRP protokolu



Obrázek 6.2: Model směrovače s protokolem EIGRP

Současná implementace podporuje jen jeden EIGRP proces na směrovači. Z časových důvodů nemohl být realizovaný návrh z předchozí kapitoly, kde toto řeší `EigrpAsSplitter`. Ze stejných důvodů není podporovaný protokol IPv6, ale pouze IPv4. S ostatními protokoly na síťové vrstvě (např. AppleTalk) jsem nepočítal, jelikož je knihovna INET neobsahuje.

Nebyla implementována autentizace, která není pro účel simulace potřeba. Rovněž funkce pro řízené využití šířky pásma rozhraní EIGRP zprávami (Packet Pacing) není součástí řešení. Z hlediska simulace není tato funkce zajímavá.

6.2 Třídy pro uchování dat

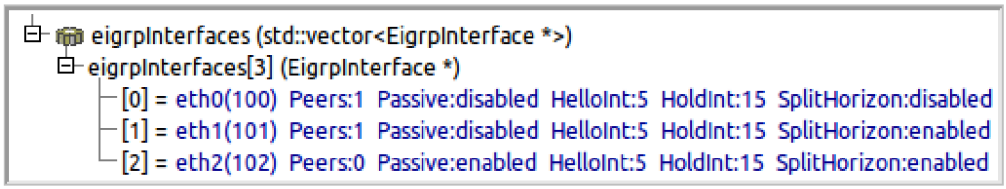
V této kapitole budou popsány moduly a třídy, které slouží primárně k uložení dat. Moduly zde uvedené nejsou spojeny s okolím, ale přistupuje se k nim voláním metod. Jejich realizace byla provedena s ohledem na možné rozšíření v podobě podpory protokolu IPv6.

6.2.1 Tabulka rozhraní EigrpInterfaceTable

Tabulka EIGRP rozhraní `EigrpInterfaceTable` byla vytvořena jako samostatný jednoduchý modul, se kterým se komunikuje voláním metod. Umožňuje provádět klasické operace, jako vyhledávání, vkládání a odebírání rozhraní. Rozhraní je zastoupeno třídou `EigrpInterface` a obsahuje nastavení z konfiguračního souboru a také nastavený *Hello* časovač. Jsou

zde pouze rozhraní, na kterých je EIGRP povoleno. Aby nebylo ztraceno nastavení protokolu např. při pádu rozhraní, jsou odebraná rozhraní přesouvána do modulu `EigrpIpv4Pdm`. K tabulce lze přistoupit pomocí vytvořené třídy `EigrpIfTableAccess` a její metody `get`.

Tabulku je možné zobrazit v simulaci, její podoba je na obr. 6.3.

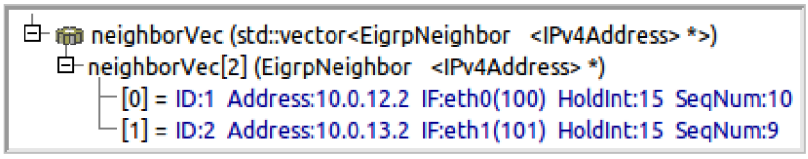


Obrázek 6.3: Podoba tabulky EIGRP rozhraní v OMNeT++

6.2.2 Tabulka sousedů EigrpIpv4NeighborTable

Dalším jednoduchým modulem je tabulka sousedů `EigrpIpv4NeighborTable`. Položku tabulky tvoří třída `EigrpNeighbor`, která se musí parametrizovat třídou pro IP adresu. Jsou zde informace o sousedovi a také nastavený časovač *Hold*. K modulu se přistupuje pomocí třídy `Eigrpv4NeighTableAccess`. Opět tabulka poskytuje základní metody pro vyhledávání, procházení, vkládání a odebírání záznamů.

Obsah tabulky je možné zobrazit při simulaci, viz obr. 6.4. Oproti výstupu příkazu `show ip eigrp neighbors` z reálného zařízení zde není zobrazena aktuální hodnota časovače *Hold*, stejně jako některé informace pro spolehlivý přenos zpráv (SRTT, RTO).



Obrázek 6.4: Grafická podoba tabulky sousedů

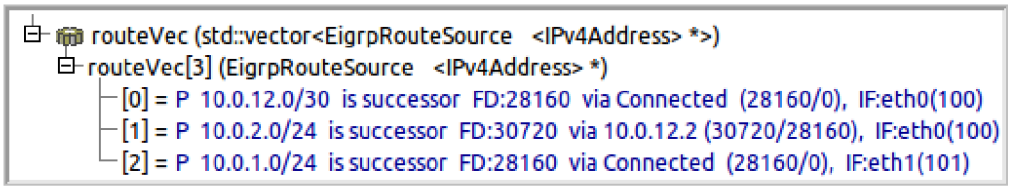
6.2.3 Tabulka topologie EigrpIpv4TopologyTable

Posledním jednoduchým modulem určeným pro uchování dat je tabulka topologie `EigrpIpv4TopologyTable`. Modul lze získat pomocí třídy `Eigrpv4TopolTableAccess`.

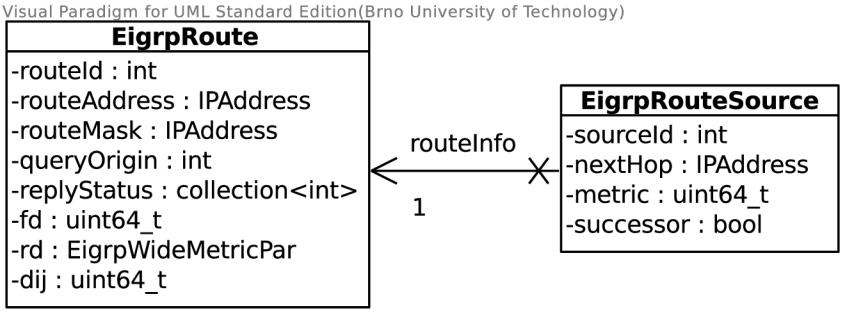
Tabulka obsahuje vektor `routeVec` s cestami a také vektor síti `routeInfoVec`. Cesta tvořená třídou `EigrpRouteSource` obsahuje informace, které se pro každou cestu odlišují. Zatímco třída `EigrpRoute` pro cíl cesty sestavá z informací společných pro všechny cesty do dané sítě. Úprava informací v `EigrpRoute` je tak snazší, než kdyby vše bylo v jedné třídě. Obě třídy jsou na obr. 6.6 s jejich nejdůležitějšími atributy. Třídy jsou parametrizovatelné typem IP adresy.

Modul poskytuje metody pro nalezení následníka (Successor), nalezení cesty s nejnižší vzdáleností, hledání FS a metody pro získání, přidání a odebrání cest a sítí.

Na obr. 6.5 je tabulka topologie tak, jak ji lze zobrazit při simulaci. První písmeno ve výpisu označuje stav cesty, tzn. ”A”pro aktivní stav, ”P”pro stav pasivní. V tomto formátu vypisují obsah tabulky i Cisco směrovače.



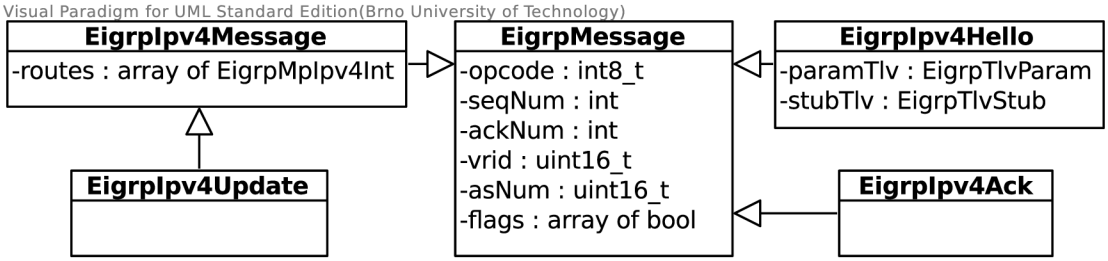
Obrázek 6.5: Tabulka topologie v OMNeT++



Obrázek 6.6: Diagram tříd pro EIGRP cestu

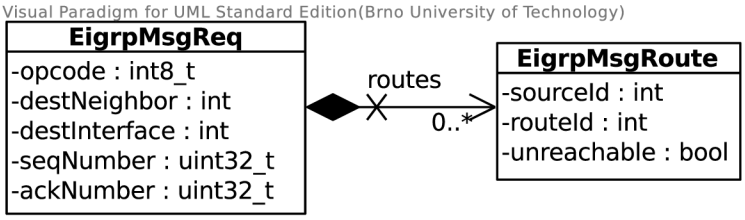
6.2.4 EIGRP zprávy

Zprávy jsou definovány v souboru `EigrpMessage.msg`, ze kterého OMNeT++ dokáže vygenerovat C++ třídy. Pro každý typ zprávy je zde třída zvlášť, a to i pro zprávu *Ack*. Hierarchii tříd lze vidět v diagramu na obrázku 6.7. Zprávy *Query*, *Update* a *Reply* nesoucí cesty mají společnou bázovou třídu `EigrpIpv4Message`, aby bylo možné s nimi pracovat v kódu jednotně.



Obrázek 6.7: Diagram tříd pro EIGRP zprávy

Pokud je třeba odeslat EIGRP zprávu, pak je nejdříve nutné vytvořit požadavek zastoupený třídou `EigrpMsgReq`. Jedná se o prostředek, pomocí kterého mezi sebou komunikují moduly `EigrpIpv4Pdm` a `EigrpRtp`. Jak je vidět z obr. 6.8, obsahuje pouze identifikátory na části zprávy, ne přímo data. Struktura `EigrpMsgRoute` obsahuje identifikátory cesty, nejedná se o TLV cesty. V okamžiku odeslání zprávy do sítě je z požadavku vytvořena EIGRP zpráva a daty.



Obrázek 6.8: Třída pro požadavek na odeslání EIGRP zprávy

6.2.5 EIGRP časovače

Implementace EIGRP využívá dva časovače, a to *Hello* a *Hold* časovač. V OMNeT++ jsou časovače realizovány jako zprávy, které modul pošle sám sobě se zadaným zpožděním. Oba časovače jsou definovány v jazyce NED v souboru `EigrpTimer.msg` a využívají stejnou třídu `EigrpTimer`. Ta obsahuje jednu položku, a to jeho druh. K časovači lze připojit ukazatel na libovolný objekt. Pro *Hello* je to odkaz na EIGRP rozhraní, časovač *Hold* je spojený s objektem, který zastupuje souseda.

6.3 Modul EigrpIpv4Pdm

Tento modul je prostředník mezi ostatními částmi EIGRP a sítí. Přehled jeho atributů a metod lze nalézt v příloze C. Přijímá zprávy ze síťové vrstvy a zpracovává je v metodě `handleMessageFromNetwork`. V této metodě rovněž dojde k přeposlání zprávy do modulu `EigrpRtp`.

Požadavky zastoupené třídou `EigrpMsgReq` tvoří rozhraní mezi tímto modulem a RTP. `EigrpIpv4Pdm` vytváří tyto požadavky a odesílá je do RTP. Odtud je i přijímá, zpracovává je v metodě `handleMessageFromRtp` a vytváří z nich EIGRP zprávy, které následně odešle do sítě.

Tento modul pracuje se třídou `EigrpDual`, jenž zastupuje automat DUAL. Třída má jedinou veřejnou metodu pro předávání informací o událostech, a to `processEvent`. Informace, které ke své činnosti automat vyžaduje při volání metody, je druh události z výčtu `DualEvent`, cesta zastoupená třídou `EigrpRouteSource` a identifikátor souseda, jenž je zdrojem události. Stav automatu je uchováván zvlášť pro každou cestu ve třídě `EigrpRoute`.

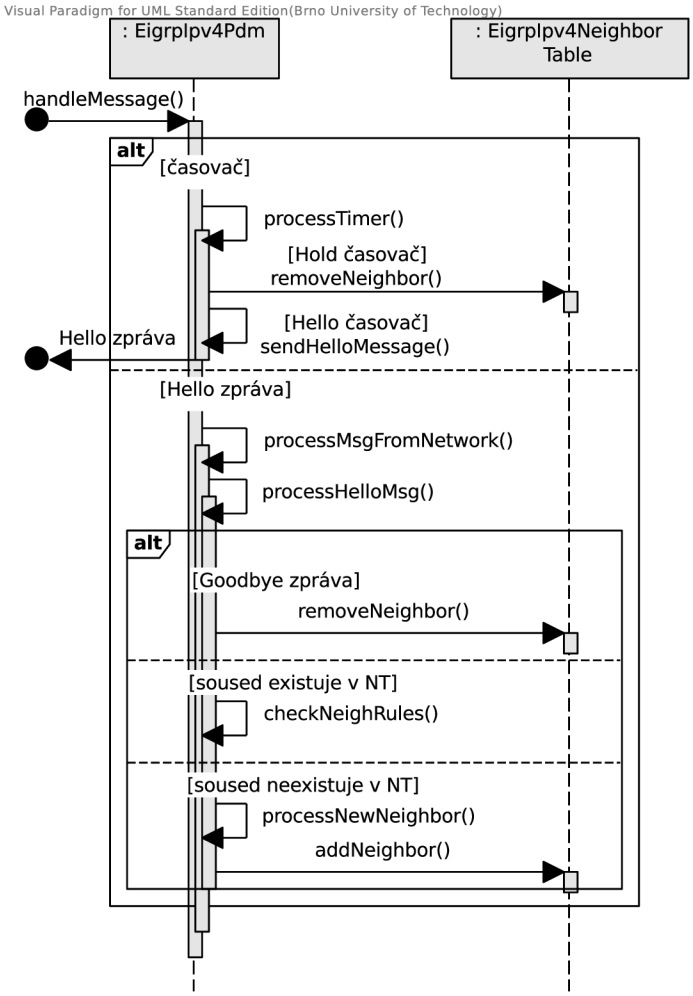
Modul má na starosti udržování sousedství, a to podle návrhu uvedeného v předchozí kapitole. Na obrázku 6.9 je ukázána implementace, kde je vidět udržování sousedství periodickým zasíláním *Hello* zpráv a také ukončení vztahu sousedství při porušení podmínek sousedství nebo příjmu zprávy Goodbye. Po povolení EIGRP na rozhraní je naplánováno odeslání první *Hello* zprávy za náhodně zvolený čas v intervalu 0 až 1 sekunda.

6.3.1 Zpracování notifikací

Součástí inicializace `EigrpIpv4Pdm` je přihlášení se k odběru notifikací z modulu s názvem `NotificationBoard`. Je tak možné reagovat na událost změny stavu rozhraní, změny konfigurace rozhraní a smazání cesty ze směrovací tabulky. Jejich zpracování řídí metoda `receiveChangeNotification`, která podle typu události předává řízení dále. Odebrání cesty má na starosti metoda `processRTRouteDel`, pro obsloužení vypnutí nebo zapnutí

rozhraní se využívá `processIfaceStateChange` a pro reakci na změnu parametrů rozhraní `processIfaceConfigChange`.

Zpracování zapnutí rozhraní je náročnější, jelikož může zahrnovat kromě události zapnutí rozhraní i smazání EIGRP cesty ve směrovaci tabulace. V diagramu 6.10 je znázorněn postup zpracování, který odpovídá návrhu v kapitole 5.3.1.



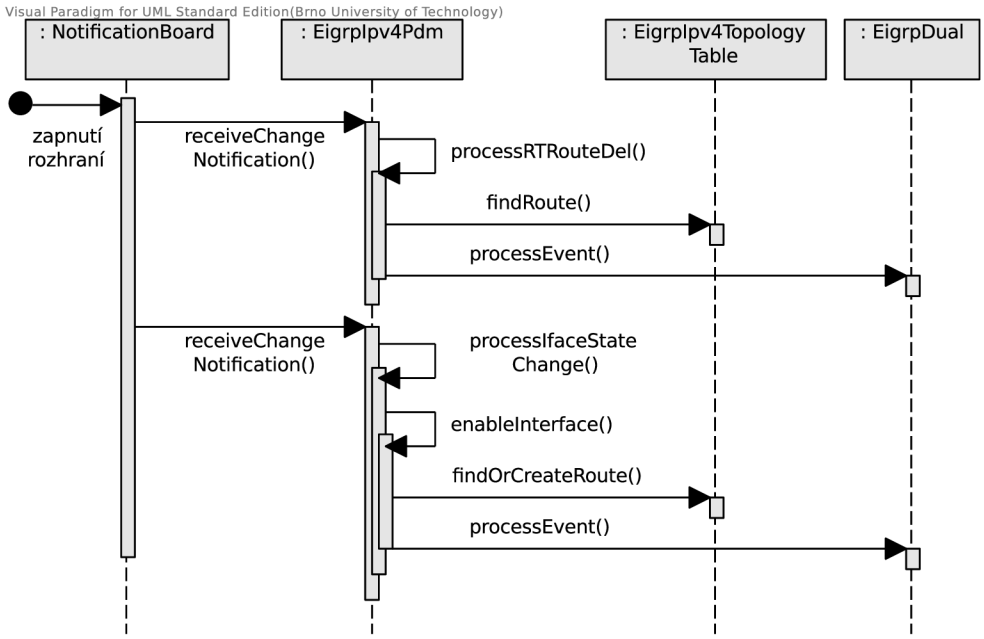
Obrázek 6.9: Třída pro požadavek na odeslání EIGRP zprávy

6.4 Modul EigrpRtp

Tento modul zajišťuje spolehlivý přenos zpráv. Spravuje frontu zpráv zastoupenou třídou `EigrpRequestQueue`, která je z důvodu optimalizace globální pro všechny sousedy. Ovšem na logické úrovní funguje, jako kdyby každý soused měl svoji. Do fronty jsou vkládány zprávy typu `EigrpMsgReq`.

Při odesílání spolehlivé zprávy se musí poznačit, kterým sousedům byla zpráva odeslána, a tedy je od nich očekáváno potvrzení. Třída `EigrpNeighbor` k tomu účelu obsahuje atribut `waitForAck` pro uložení očekávaného čísla potvrzení.

V současné implementaci není podporováno znovuzaslání zprávy po vypršení doby RTO.



Obrázek 6.10: Zpracování zapnutí rozhraní

6.5 Konfigurace EIGRP modulu

Jedním z výstupů projektu ANSA je modul `DeviceConfigurator`, jenž se snaží sjednotit konfiguraci všech protokolů do společného formátu XML. Jednou z výhod oproti jiným způsobům načtení konfigurace je, že dovoluje pro některé protokoly automatickou transformaci nastavení z reálného zařízení do formátu XML.

Modul `DeviceConfigurator` byl rozšířen o metody pro načtení EIGRP konfigurace. K tomu účelu jsem vytvořil metodu `loadEigrpIPv4Config`, která získá nastavení protokolu ze souboru. Nejdříve je načtena konfigurace procesu, následně nastavení jednotlivých rozhraní. Formát konfigurace je shodný s tím, který byl uvedený v kapitole 5.5.2.

Kapitola 7

Porovnání simulace a reálné sítě

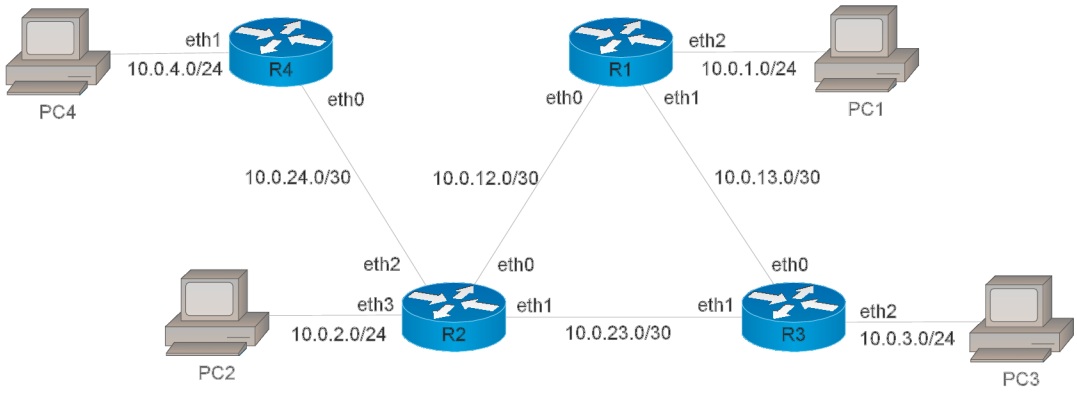
V této kapitole je porovnána implementace protokolu EIGRP v OMNeT++ oproti chování protokolu na reálných zařízeních. Pro testování byla použita zařízení Cisco C2911 s operačním systémem IOS 15.2 (4) M1 (c2900-universalk9-mz.SPA.152-4.M1.bin).

Posloupnosti zpráv a časová razítka patřící k průběhům v reálné topologii byly zijšťovány z ladících výpisů protokolu EIGRP (příkaz *debug eigrp packets*) a odchycené komunikace pomocí programu Wireshark.

Při testování je nutné brát v úvahu vlastnosti simulátoru, a to především nulovou dobu zpracování událostí, kde veškeré zpoždění je tvořeno přenosem zpráv. Dále EIGRP v OMNeT++ nemá implementovaný mechnismus pro zpožděné odesílání zpráv, které zaručuje zatížení linky EIGRP provozem jen na určité procento šířky pásma.

Čas T_0 uvedený v testech se vždy vztahuje k nějaké události, která bude popsána. Časové rozdíly mezi událostmi v simulaci jsou v některých případech velmi malé. Pokud je rozdíl menší než jedna tisícina sekundy, pak je časový rozdíl zvýšen na jednu tisícinu.

Ve scénářích, které následují, jsou obvykle uvedeny výstupy z jednoho zařízení. Důvodem je velké množství informací, které by text zahltili. Obsahy všech tabulek z reálné sítě byly porovnány vůči simulaci, případné odchylky jsou uvedeny. Také směrovací tabulky nejsou součástí textu, jelikož jsou tvořeny na základě tabulek topologie, a tedy by bylo zbytečné vypisovat i jejich obsah. Všechny výstupy lze nalézt v příloze A.



Obrázek 7.1: Topologie testované sítě

7.1 Test ustavení sousedství

V tomto testu je ukázán proces vytvoření sousedství mezi směrovači a výměna cest. V simulaci byly všechny směrovače zapnuty ve stejnou chvíli. Na reálných zařízeních se toho však dosahuje těžko, takže obsah zpráv nemusí být úplně shodný se simulací. Ovšem výsledná tabulka topologie odpovídá. Na obrázku 7.1 je topologie sítě, na které byl test proveden.

Fáze	**Typ zprávy**	**Směr**	**Čas v sim. [s]**	**Čas v real. [s]**
1	*Hello*	R1 → R2	0.548	1.572
1	*Hello*	R2 → R1	0.549	1.575
1	*Hello*	R1 → R2	0.550	1.576
2	*Update* (INIT)	R2 → R1	0.551	1.579
2	*Update* (INIT)	R1 → R2	0.552	1.580
3	*Update*	R2 → R1	0.554	1.583
3	*Update*	R1 → R2	0.553	1.592
3	*Update*	R1 → R2	0.556	1.600
3	*Update*	R2 → R1	0.555	1.691

Tabulka 7.1: Čas přenosu zpráv pro ustavení sousedství v reálné síti a v simulaci

Průběh testu lze rozdělit do několika fází:

1. Nejdříve dojde k odesílání *Hello* zpráv pro navázání sousedství;
2. Následuje odeslání zprávy *Update* s příznakem INIT, což je žádost o zaslání všech směrovacích informací;
3. Nakonec odeslání jedné nebo více zpráv *Update* s cestami.

V tabulce 7.1 jsou uvedeny zprávy přenášené při vytvoření sousedství mezi směrovači *R1* a *R2* na reálných zařízeních a v simulaci. Čas T_0 je okamžik povolení EIGRP procesu na směrovačích. Na obrázku 7.3 je obsah tabulky topologie na *R2*, na obrázku 7.2 je tabulka sousedů ze stejného směrovače. Tabulky byly získány až po ustavení sousedství mezi všemi sousedními směrovači.

```
H   Address      Interface  Hold Uptime   SRTT   RTO  Q  Seq
                            (sec)         (ms)       Cnt Num
2   10.0.24.2    Gi0/2      11 00:04:26    1     100  0  24
1   10.0.23.2    Gi0/1      12 00:04:30    1     505  0  61
0   10.0.12.1    Gi0/0      12 00:04:31    1     225  0  62
```

(a) Reálná síť

```
[0] = ID:1 Address:10.0.23.2 IF:eth1(101) HoldInt:15 SeqNum:10
[1] = ID:2 Address:10.0.12.1 IF:eth0(100) HoldInt:15 SeqNum:8
[2] = ID:3 Address:10.0.24.2 IF:eth2(102) HoldInt:15 SeqNum:3
```

(b) Simulace

Obrázek 7.2: Tabulka sousedů na směrovači R2

Následuje několik poznámek k průběhu testu.

- Jak je vidět, na reálných zařízeních je prodleva mezi spuštěním EIGRP procesu na směrovači a odesláním první *Hello* zprávy několik sekund. Vysvětluji si to zpožděním

```
P 10.0.3.0/24, 1 successors, FD is 30720, serno 57
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.1.0/24, 1 successors, FD is 30720, serno 54
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (33280/30720), GigabitEthernet0/1
P 10.0.2.0/24, 1 successors, FD is 28160, serno 50
        via Connected, Loopback0
P 10.0.13.0/30, 2 successors, FD is 30720, serno 58
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
P 10.0.4.0/24, 1 successors, FD is 30720, serno 59
        via 10.0.24.2 (30720/28160), GigabitEthernet0/2
P 10.0.23.0/30, 1 successors, FD is 28160, serno 52
        via Connected, GigabitEthernet0/1
P 10.0.12.0/30, 1 successors, FD is 28160, serno 51
        via Connected, GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 28160, serno 53
        via Connected, GigabitEthernet0/2
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[2] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[3] = P 10.0.1.0/24 FD:30720 via 10.0.23.2 (33280/30720), IF:eth1(101)
[4] = P 10.0.3.0/24 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[5] = P 10.0.3.0/24 FD:30720 via 10.0.12.1 (33280/30720), IF:eth0(100)
[6] = P 10.0.13.0/30 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[7] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[8] = P 10.0.2.0/24 is successor FD:28160 via Connected (28160/0), IF:eth3(103)
[9] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[10] = P 10.0.23.0/30 is successor FD:28160 via Connected (28160/0), IF:eth1(101)
```

(b) Simulace

Obrázek 7.3: Obsah tabulky topologie na R2

způsobeném zavedením EIGRP procesu, jelikož EIGRP pro toto nedefinuje žádný časovač. V simulaci je odeslání první *Hello* zprávy naplánované na náhodně zvolený čas v rozmezí 0 až 1 sekund.

- Jiné pořadí při odesílání zpráv *Update* mezi *R1* a *R2* v simulaci ve fázi 3 je způsobno tím, že v simulaci směrovač *R1* nevloží do zprávy *Update* s INIT příznakem číslo potvrzení, jelikož ještě nepřijal *Update* od *R2*. V reálné síti *R1* před odesláním *Update* s INIT přijme od *R2* *Update* s INIT příznakem. Díky tomu může *R1* vložit do zprávy *Update* číslo potvrzení. Směrovač *R2* pak přijme *Update* s INIT a číslem potvrzení a ihned může odeslat *Update* s cestami. Nemusí tedy čekat na zprávu *Ack* jako při simulaci.

7.2 Test pádu rozhraní

Pro stejnou topologii po ustavení sousedství a konvergenci sítě je na směrovači *R2* vypnuto rozhraní *eth1* k *R3*. V tabulce 7.2 jsou zachyceny zprávy z reálných zařízení a ze simulátoru. Jsou zde uvedeny podstatné zprávy ze všech směrovačů, jelikož je žádoucí zachytit akce prováděné na *R2* a také na *R3*. T_0 je čas vypnutí rozhraní. Následuje stručný popis k přenášeným zprávám.

1. Cesty 10.0.3.0/24 a 10.0.23.0/30 se na *R2* staly nedostupné, a tak je pro ně započat difúzní výpočet a jsou odeslány zprávy *Query*. Poté je odeslána zpráva *Update* na *R1* a *R4*, jelikož směrovač smazal cestu do sítě 10.0.13.0/30 přes *R3* ve směrovací tabulce;

Fáze	Typ zprávy	Směr	Čas v sim. [s]	Čas v real. [s]
1	*Query*	R2 → R4	T_0	0.016
1	*Query*	R2 → R1	T_0	0.016
1	*Update*	R2 → R1	0.001	0.023
1	*Update*	R2 → R4	0.001	0.023
2	*Reply*	R1 → R2	0.002	0.037
2	*Reply*	R4 → R2	0.002	0.035
2	*Update*	R1 → R2	0.002	0.045
3	*Update*	R2 → R1	0.003	0.054
3	*Update*	R2 → R4	0.003	0.057
4	*Update*	R4 → R2	0.004	0.075
5	*Hello*	R3 → R2	11.253	11.245
5	*Query*	R3 → R1	11.253	11.250
5	*Update*	R3 → R1	11.255	11.258
6	*Reply*	R1 → R3	11.254	11.269
6	*Update*	R3 → R1	11.256	11.286

Tabulka 7.2: Čas přenosu zpráv v reálné síti a v simulaci při pádu rozhraní

2. *R4* přijme *Query*, ale nenalezne FS. Jelikož má jediného souseda a pravidlo Split Horizon je zapnuté, nemá komu odeslat *Query*. Odešle proto v *Reply* informaci o nedostupnosti cest. Směrovač *R1* také přijme zprávu *Query*, ale nalezne FS pro cesty z *Query* a odešle je ve zprávě *Reply* zpět. Navíc ze zprávy zjistí, že cesta do sítě 10.0.23.0/30 je přes *R2* nedostupná a smaže ji ze směrovací tabulky. O této změně informuje ostatní pomocí *Update*;

3. *R2* přijme poslední *Reply* a vybere směrovač *R1* jako nového následníka (Successor). Upraví směrovací tabulku a odešle sousedům *Update*;

4. Směrovač *R4* přijme *Update* a upraví své záznamy ve směrovací tabulce. Odešle zprávu *Update* zpět s informací o nedostupnosti cest přes něj. *R1* také přijme *Update* od *R2*, neprovede žádné akce;

5. *R3* dosud nezaznamenal změnu topologie. Za přibližně 11 sekund na směrovači *R3* vyprší časovač *Hold*, a proto *R3* ukončí sousedství s *R2*. Na rozhraní *eth1* k *R2* odešle zprávu Goodbye. Dále pro všechny sítě dostupné přes *R2* spustí difúzní výpočet a odešle *Query* na *R1*. Také odešle zprávu *Update*, jelikož ze směrovací tabulky odstranil cestu skrze *R2* do sítě 10.0.12.0/30;

6. *R1* přijme *Query*, nalezne FS a odešle je v *Reply*. *R3* zpracuje *Reply*, upraví záznamy ve směrovací tabulce a odešle zpět *Update* (Poison Reverse).

```
[0] = ID:2 Address:10.0.12.1 IF:eth0(100) HoldInt:15 SeqNum:12
[1] = ID:3 Address:10.0.24.2 IF:eth2(102) HoldInt:15 SeqNum:5
```

Obrázek 7.4: Tabulka sousedů na směrovači R2 ze simulátoru

```
P 10.0.3.0/24, 1 successors, FD is 33280, serno 64
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.1.0/24, 1 successors, FD is 30720, serno 54
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
P 10.0.2.0/24, 1 successors, FD is 28160, serno 50
        via Connected, Loopback0
P 10.0.13.0/30, 1 successors, FD is 30720, serno 62
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
P 10.0.4.0/24, 1 successors, FD is 30720, serno 59
        via 10.0.24.2 (30720/28160), GigabitEthernet0/2
P 10.0.23.0/30, 1 successors, FD is 33280, serno 63
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.12.0/30, 1 successors, FD is 28160, serno 51
        via Connected, GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 28160, serno 53
        via Connected, GigabitEthernet0/2
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[2] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[3] = P 10.0.3.0/24 is successor FD:33280 via 10.0.12.1 (33280/30720), IF:eth0(100)
[4] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[5] = P 10.0.2.0/24 is successor FD:28160 via Connected (28160/0), IF:eth3(103)
[6] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[7] = P 10.0.23.0/30 is successor FD:33280 via 10.0.12.1 (33280/30720), IF:eth0(100)
```

(b) Simulace

Obrázek 7.5: Obsah tabulky topologie na R2

Obsah tabulky topologie ze směrovače *R2* po zpracování událostí, které souvisejí s vypnutím rozhraní, je na obr. 7.5. Jsou zvýrazněny cesty, které původně vedli přes směrovač *R3*. Ovšem po pádu rozhraní byl vybrán jako nejlepší následník směrovač *R1*. Na obr. 7.4 je tabulka sousedů směrovače *R2* v simulátoru, ze které byl odebrán soused *R3*.

Z tabulky se zprávami jde vidět, že ve fázi 5 a 6 v simulaci předběhne zpráva *Update* odeslaná z *R3* zprávu *Reply* od *R1*. Je to způsobeno dobou zpracování cest z *Reply*, která je na reálném zařízení nenulová.

7.3 Test zapnutí rozhraní

Tento scénář v návaznosti na předchozí test ověřuje zapnutí rozhraní *eth1* na směrovači *R2*. Posloupnost zpráv z reálné sítě a ze simulace je v 7.3. Jsou uvedeny jen zprávy týkající se směrovače *R2*.

Fáze	**Typ zprávy**	**Směr**	**Čas v sim. [s]**	**Čas v real. [s]**
1	*Query*	R2 → R4	T_0	0.016
1	*Query*	R2 → R1	T_0	0.016
2	*Reply*	R1 → R2	0.001	0.034
2	*Reply*	R4 → R2	0.001	0.036
2	*Update*	R2 → R1	0.002	0.051
2	*Update*	R2 → R4	0.002	0.052
2	*Update*	R1 → R2	0.003	0.066
2	*Update*	R4 → R2	0.003	0.068

Tabulka 7.3: Výměna zpráv mezi sousedy při zapnutí rozhraní na R2

Nejdůležitější akce po zapnutí rozhraní jsou následující. Zapnutí rozhraní způsobí, že se vloží na *R2* do směrovací tabulky nová přímo připojená cesta. To také znamená, že se smaže v této tabulce EIGRP cesta do sítě 10.0.23.0/32. DUAL zaznamená odebrání cesty ze směrovací tabulky a vyvolá ztrátu všech cest do této sítě. Následuje přechod do aktivního stavu a odeslání *Query*. V tomto okamžiku je DUAL na *R2* upozorněn na zapnutí rozhraní a je mu předána přímo připojená cesta. Sousedé odpoví na *Query* od *R2* zprávami *Reply* a směrovač *R2* informuje všechny sousedy o výsledku difúzního výpočtu pomocí zprávy *Update*. *R1* a *R4* vloží cestu z přijaté zprávy *Update* do směrovací tabulky a odešlou zpět na *R2* *Update* (Poison Reverse). Po několika sekudnách přijme *R3* zprávu *Hello* od směrovače *R2* a začne klasický proces ustavení sousedství, který byl již popsán. Zprávy pro ustavení sousedství v tabulce nejsou zobrazeny.

Obsah tabulky topologie ze směrovače *R2* po zkonvergování sítě je na obr. 7.6. Je vidět, že některé cesty vedou přes *R3*, jelikož mají kratší vzdálenost do cíle. Tabulka sousedů je shodná s tabulkou uvedenou v testu pro ustavení sousedství, viz 7.1.

```
P 10.0.3.0/24, 1 successors, FD is 30720, serno 67
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.1.0/24, 1 successors, FD is 30720, serno 54
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (33280/30720), GigabitEthernet0/1
P 10.0.2.0/24, 1 successors, FD is 28160, serno 50
        via Connected, Loopback0
P 10.0.13.0/30, 2 successors, FD is 30720, serno 68
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
P 10.0.4.0/24, 1 successors, FD is 30720, serno 59
        via 10.0.24.2 (30720/28160), GigabitEthernet0/2
P 10.0.23.0/30, 1 successors, FD is 28160, serno 66
        via Connected, GigabitEthernet0/1
P 10.0.12.0/30, 1 successors, FD is 28160, serno 51
        via Connected, GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 28160, serno 53
        via Connected, GigabitEthernet0/2
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[2] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[3] = P 10.0.1.0/24 FD:30720 via 10.0.23.2 (33280/30720), IF:eth1(101)
[4] = P 10.0.3.0/24 FD:30720 via 10.0.12.1 (33280/30720), IF:eth0(100)
[5] = P 10.0.3.0/24 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[6] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[7] = P 10.0.13.0/30 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[8] = P 10.0.2.0/24 is successor FD:28160 via Connected (28160/0), IF:eth3(103)
[9] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[10] = P 10.0.23.0/30 is successor FD:28160 via Connected (28160/0), IF:eth1(101)
```

(b) Simulace

Obrázek 7.6: Obsah tabulky topologie na R2

7.4 Test vypnutí Split Horizon

Na stejné topologii je ověřeno chování EIGRP při vypnutém pravidlu Split Horizon na rozhraní *eth0* směrovače *R1*, tj. směrem k *R2*. Na *R2* je na rozhraní *eth3* k *PC2* naplánovaná změna zpoždění ze 100 na 110 mikrosekund.

V tabulce 7.4 je posloupnost zpráv přenášených mezi sousedy po změně zpoždění. Čas T_0 je okamžik změny parametru rozhraní. Ve fázi jedna směrovač *R2* odešle *Update* všem

Fáze	Typ zprávy	Směr	Čas v sim. [s]	Čas v real. [s]
1	*Update*	R2 → R4	T_0	0.016
1	*Update*	R2 → R1	T_0	0.016
1	*Update*	R2 → R3	T_0	0.016
2	*Update*	R3 → R1	0.001	0.030
2	*Update*	R1 → R2	0.001	0.033
2	*Update*	R1 → R3	0.001	0.033

Tabulka 7.4: Výměna zpráv mezi sousedy při vypnutém Split Horizon

sousedům s informací o změně metriky. Ve druhé fázi sousedé zpracují *Update* od *R2*, upraví tabulku topologie a směrovací tabulku. Následně chtějí odeslat *Update* všem svým sousedům. Při odesílání zprávy směrovače *R3* a *R4* aplikují pravidlo Split Horizon, a tak ji neodešlou na *R2*. Ovšem na *R1* je pravidlo vypnuté, takže tuto zprávu na *R2* odešle.

Obsah tabulky topologie ze směrovače *R2* je na obr. 7.7. Kromě zvýšení metriky cesty do sítě 10.0.2.0/24 je vidět, že *R2* má oproti stavu se zapnutým Split Horizon navíc dvě cesty. Jedná se o cestu do sítě 10.0.12.0/30 a 10.0.4.0/24, obě přes *R1*. Jsou zde z toho důvodu, že při ustavení sousedství nebylo na směrovači *R1* na rozhraní k *R2* aplikováno pravidlo Poison Reverse, které je spjaté se Split Horizon.

```
P 10.0.3.0/24, 1 successors, FD is 30720, serno 177
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.1.0/24, 1 successors, FD is 30720, serno 173
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (33280/30720), GigabitEthernet0/1
P 10.0.2.0/24, 1 successors, FD is 28416, serno 179
        via Connected, Loopback0
P 10.0.13.0/30, 2 successors, FD is 30720, serno 178
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
P 10.0.4.0/24, 1 successors, FD is 30720, serno 176
        via 10.0.24.2 (30720/28160), GigabitEthernet0/2
        via 10.0.12.1 (35840/33280), GigabitEthernet0/0
P 10.0.23.0/30, 1 successors, FD is 28160, serno 171
        via Connected, GigabitEthernet0/1
P 10.0.12.0/30, 1 successors, FD is 28160, serno 170
        via Connected, GigabitEthernet0/0
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 28160, serno 172
        via Connected, GigabitEthernet0/2
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[2] = P 10.0.4.0/24 FD:30720 via 10.0.12.1 (35840/33280), IF:eth0(100)
[3] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[4] = P 10.0.1.0/24 FD:30720 via 10.0.23.2 (33280/30720), IF:eth1(101)
[5] = P 10.0.12.0/30 FD:28160 via 10.0.12.1 (30720/28160), IF:eth0(100)
[6] = P 10.0.3.0/24 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[7] = P 10.0.3.0/24 FD:30720 via 10.0.12.1 (33280/30720), IF:eth0(100)
[8] = P 10.0.13.0/30 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[9] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[10] = P 10.0.2.0/24 is successor FD:28416 via Connected (28416/0), IF:eth3(103)
[11] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[12] = P 10.0.23.0/30 is successor FD:28160 via Connected (28160/0), IF:eth1(101)
```

(b) Simulace

Obrázek 7.7: Tabulka topologie ze směrovače R2 při vypnutém Split Horizon

7.5 Test nestejnoměrného vyvažování zátěže

Je využita stejná síť, jako v předchozích testech. Pro ověření nestejnoměrného vyvažování zátěže bylo na směrovači *R3* na rozhraní *eth0* k *R1* nastaven parametr zpoždění (Delay) na 110 mikrosekund. Dále na *R2* byl nastaven parametr *variance* na hodnotu 2.

Obsah tabulky topologie po ustavení sousedství mezi všemi směrovači je na obr. 7.8. Směrovač *R2* má díky nestejnoměrnému vyvažování v tabulce topologie dvě cesty do sítě 10.0.13.0/30 označené jako následníky (Successor), a to s různou metrikou. Reálný směrovač má oproti simulaci do sítě s adresou 10.0.1.0/24 také dva následníky, ačkoli cesta přes 10.0.23.2 by takto označena být neměla. Nesplňuje nutnou podmínku Feasibility Condition [24]. To samé platí pro síť 10.0.3.0/24 a cestu přes 10.0.12.1.

```
P 10.0.3.0/24, 2 successors, FD is 30720, serno 87
        via 10.0.23.2 (30720/28160), GigabitEthernet0/1
        via 10.0.12.1 (33280/30720), GigabitEthernet0/0
P 10.0.1.0/24, 2 successors, FD is 30720, serno 88
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (33536/30976), GigabitEthernet0/1
P 10.0.2.0/24, 1 successors, FD is 28160, serno 79
        via Connected, Loopback0
P 10.0.13.0/30, 2 successors, FD is 30720, serno 86
        via 10.0.12.1 (30720/28160), GigabitEthernet0/0
        via 10.0.23.2 (30976/28416), GigabitEthernet0/1
P 10.0.4.0/24, 1 successors, FD is 30720, serno 89
        via 10.0.24.2 (30720/28160), GigabitEthernet0/2
P 10.0.23.0/30, 1 successors, FD is 28160, serno 81
        via Connected, GigabitEthernet0/1
P 10.0.12.0/30, 1 successors, FD is 28160, serno 80
        via Connected, GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 28160, serno 82
        via Connected, GigabitEthernet0/2
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[2] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[3] = P 10.0.1.0/24 FD:30720 via 10.0.23.2 (33536/30976), IF:eth1(101)
[4] = P 10.0.3.0/24 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[5] = P 10.0.3.0/24 FD:30720 via 10.0.12.1 (33280/30720), IF:eth0(100)
[6] = P 10.0.13.0/30 is successor FD:30720 via 10.0.23.2 (30976/28416), IF:eth1(101)
[7] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[8] = P 10.0.2.0/24 is successor FD:28160 via Connected (28160/0), IF:eth3(103)
[9] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[10] = P 10.0.23.0/30 is successor FD:28160 via Connected (28160/0), IF:eth1(101)
```

(b) Simulace

Obrázek 7.8: Tabulka topologie ze směrovače R2

7.6 Test vypnutí vyvažování zátěže

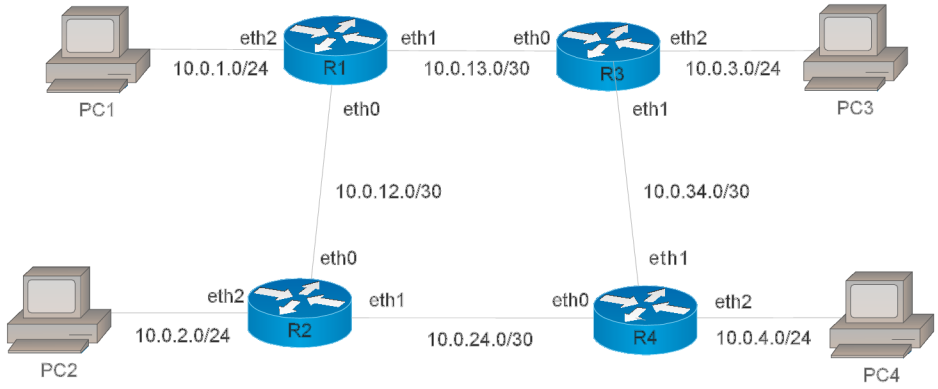
Topologie sítě je shodná s předchozími scénáři. Na *R2* je vypnuto vyvažování zátěže. Na obrázku 7.9 je obsah tabulky topologie na směrovači *R2* v simulátoru po ustavení sousedství mezi všemi sousedy. Díky vypnutému vyvažování není cesta do sítě 10.0.13.0/32 přes 10.0.23.2 (tj. *R3*) v tabulce topologie označena jako následník, a tedy není vložena do směrovací tabulky.

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.3.0/24 is successor FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[2] = P 10.0.3.0/24 FD:30720 via 10.0.12.1 (33280/30720), IF:eth0(100)
[3] = P 10.0.1.0/24 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[4] = P 10.0.1.0/24 FD:30720 via 10.0.23.2 (33280/30720), IF:eth1(101)
[5] = P 10.0.13.0/30 is successor FD:30720 via 10.0.12.1 (30720/28160), IF:eth0(100)
[6] = P 10.0.13.0/30 FD:30720 via 10.0.23.2 (30720/28160), IF:eth1(101)
[7] = P 10.0.4.0/24 is successor FD:30720 via 10.0.24.2 (30720/28160), IF:eth2(102)
[8] = P 10.0.2.0/24 is successor FD:28160 via Connected (28160/0), IF:eth3(103)
[9] = P 10.0.24.0/30 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
[10] = P 10.0.23.0/30 is successor FD:28160 via Connected (28160/0), IF:eth1(101)
```

Obrázek 7.9: Tabulka topologie na R2 v simulaci

7.7 Test stub směrování

V tomto testu je ověřena funkce stub směrování na topologii z obrázku 7.10. *R2* a *R4* jsou nakonfigurovaní jako stub směrovače a šíří sítí pouze cesty přímo připojené nebo sumarizované. Nejdříve je ustaveno sousedství mezi všemi sousedy. Pak na směrovači *R1* nastane pád rozhraní *eth1* k *R3*.



Obrázek 7.10: Topologie sítě pro ověření stub směrování

Následuje vysvětlení k přenášeným zprávám, jež jsou uvedeny v tabulce 7.5. *R1* po pádu rozhraní ztratí cesty do některých sítí, a proto pro ně DUAL přejde do aktivního stavu. Jelikož je jediný soused *R2* stub, nemá *R1* komu odeslat *Query*. Přejde proto zpět do pasivního stavu a odešle zprávu *Update* na *R2* s informací o nedostupnosti cest (10.0.3.0/24, 10.0.13.0/30, 10.0.34.0/30 a 10.0.4.0/24). *R2* přijme *Update* a pro některé cesty z této zprávy (10.0.3.0/24, 10.0.13.0/30) přejde do aktivního stavu a odešle *Query* na *R1* a *R4*. Směrovač *R1* cesty do sítí v *Query* nezná, a proto odešle *Reply* oznamující nedostupnost cest. Směrovač *R4* je stub, a tedy musí odpovědět na *Query* zprávou *Reply* s nekonečnou metrikou pro každou cestu z *Query*. Po přibližně 15 sekundách vyprší časovač *Hold* na *R3*. Podobně jako před chvílí *R4* přejde pro některé cesty z přijetého *Update* od *R3* do aktivního stavu a odešle *Query* všem sousedům. *R2* a *R3* následně odpoví pomocí *Reply*.

Tabulka topologie ze směrovače *R1* je na obrázku 7.11. Je vidět, že některé sítě jsou pro *R1* nedostupné. Tento směrovač tedy nevyužívá stub souseda *R2* pro dosažení ztracených sítí (např. 10.0.13.0/30), což je jedna z hlavních funkcí stub směrování.

Fáze	Typ zprávy	Směr	Čas v sim. [s]	Čas v real. [s]
1	*Update*	R1 → R2	T_0	0.016
1	*Query*	R2 → R4	0.001	0.034
1	*Query*	R2 → R1	0.001	0.034
1	*Reply*	R4 → R2	0.002	0.055
1	*Reply*	R1 → R2	0.003	0.056
2	*Update*	R3 → R4	14.910	14.911
2	*Query*	R4 → R2	14.911	14.926
2	*Query*	R4 → R3	14.911	14.926
2	*Reply*	R2 → R4	14.912	14.946
2	*Reply*	R3 → R4	14.913	14.947

Tabulka 7.5: Výměna zpráv mezi sousedy při zapnutém stub směrování

```
P 10.0.1.0/24, 1 successors, FD is 28160, serno 142
        via Connected, Loopback0
P 10.0.2.0/24, 1 successors, FD is 30976, serno 176
        via 10.0.12.2 (30976/28416), GigabitEthernet0/0
P 10.0.12.0/30, 1 successors, FD is 28160, serno 143
        via Connected, GigabitEthernet0/0
P 10.0.24.0/30, 1 successors, FD is 30720, serno 177
        via 10.0.12.2 (30720/28160), GigabitEthernet0/0
```

(a) Reálná síť

```
[0] = P 10.0.12.0/30 is successor FD:28160 via Connected (28160/0), IF:eth0(100)
[1] = P 10.0.2.0/24 is successor FD:30976 via 10.0.12.2 (30976/28416), IF:eth0(100)
[2] = P 10.0.24.0/30 is successor FD:30720 via 10.0.12.2 (30720/28160), IF:eth0(100)
[3] = P 10.0.1.0/24 is successor FD:28160 via Connected (28160/0), IF:eth2(102)
```

(b) Simulace

Obrázek 7.11: Tabulka topologie ze směrovače R2

Kapitola 8

Závěr

Simulace počítačových sítí je důležitá především pro analýzu chování a verifikaci konfigurace protokolů a zařízení. Simulační nástroj OMNeT++ spolu s knihovnou INET jsou stále ve vývoji a některé protokoly zde chybí. Cílem této práce je začlenit směrovací protokol EIGRP do architektury INET.

Je zde podrobně rozebrán protokol EIGRP a principy, na kterých stojí. Je popsán simulační nástroj OMNeT++ a knihovna INET pro simulaci počítačových sítí a možnosti pro nasazení protokolu EIGRP do jejich architektury. Zabývám se zde také konfigurací tohoto protokolu na zařízeních značky Cisco, součástí je i ověření konfigurace.

Další text je věnován návrhu rozšíření knihovny INET o protokol EIGRP. Je zde popsáno chování jednotlivých částí protokolu a jejich napojení na stávající strukturu knihovny.

Následovala implementace EIGRP. Jelikož jsem protokol navrhl jako několik nezávislých částí, byl pro něj vytvořen složený modul. Zobrazení dat v simulaci bylo přizpůsobeno vzhledu výstupů na Cisco směrovačích. Formátem pro konfiguraci protokolu je XML, jehož struktura je podobná konfiguraci EIGRP na Cisco zařízeních.

V poslední části textu byla implementace v simulátoru ověřena vůči reálné síti. Vybral jsem dvě topologie a na nich provedl celkem sedm testů, které postihují všechny implementované funkce EIGRP. Ověřují také chování protokolu při událostech, jako např. vypnutí a zapnutí rozhraní a změna parametru rozhraní pro výpočet metriky.

Rozšíření této práce by mohlo spočívat v přidání podpory pro protokol IPv6 a také možnosti používat IPv4 a IPv6 současně jedním EIGRP procesem. Zbývá dodělat některé funkce, jako např. řešení problému Stuck-In-Active a funkci pro rychlé odesílání multicastových zpráv. Dalším vhodným rozšířením je přidání sumarizace a redistribuce cest, které EIGRP rovněž umožňuje.

Literatura

[1] ANSA: Automated Network Simulation and Analysis [online]. 2012 [cit. 2014-01-05].
URL https://nes.fit.vutbr.cz/ANSA/pmwiki.php

[2] Aziz, Z.; Lui, J.; Martey, A.; aj.: Troubleshooting EIGRP [online]. Červenec 2002 [cit. 2014-01-04].
URL http://www.ciscopress.com/articles/article.asp?p=27839

[3] Cisco System, Inc.: Enhanced Interior Gateway Routing Protocol [online]. 2005 [cit. 2014-01-06].
URL http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a0080094cb7.shtml

[4] Cisco System, Inc.: Interface Commands [online]. 2006 [cit. 2014-02-05].
URL http://www.cisco.com/c/en/us/td/docs/ios/12_2/interface/command/reference/finter_r/irfshoin.html

[5] Cisco System, Inc.: Implementing EIGRP for IPv6 [online]. 2011 [cit. 2014-02-05].
URL http://www.cisco.com/en/US/docs/ios/ipv6/configuration/guide/ip6-eigrp.html

[6] Cisco System, Inc.: Advances in EIGRP [online]. 2012 [cit. 2014-02-05].
URL http://www.cisco.com/c/dam/en/us/products/collateral/ios-nx-os-software/enhanced-interior-gateway-routing-protocol-eigrp/Advances_In_EIGRP.pdf

[7] Cisco System, Inc.: Statistics Service Commands. 2013 [2014-05-06].
URL http://www.cisco.com/c/en/us/td/docs/routers/crs/software/crs_r4-2/system_monitoring/command/reference/b_sysmon_cr42crs/b_sysmon_cr42crs_chapter_0111.html

[8] Cisco System, Inc.: Cisco IOS IP Routing: EIGRP Command Reference. 2014 [2014-04-06].
URL http://www.cisco.com/c/en/us/td/docs/ios/iproute_eigrp/command/reference/ire_book.pdf

[9] Cisco System, Inc.: Cisco Nonstop Forwarding for EIGRP. [cit. 2014-01-06].
URL http://www.cisco.com/en/US/technologies/tk648/tk365/technologies_white_paper0900aecd8023df74_ps6599_Products_White_Paper.html

[10] Cisco System, Inc.: Preventing Duplicate EIGRP Router IDs [online]. Leden 2008 [cit. 2014-02-05].

URL http://www.cisco.com/c/en/us/support/docs/ip/enhanced-interior-gateway-routing-protocol-eigrp/18685-eigrp-dup-id.html

[11] Cisco System, Inc.: EIGRP Wide Metrics [online]. Prosinec 2012 [cit. 2014-02-05]. URL http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/iproute_eigrp/configuration/15-sy/ire-15-sy-book/ire-wid-met.html

[12] Cisco System, Inc.: What Does the EIGRP DUAL-3-SIA Error Message Mean? [online]. Srpen 2005 [cit. 2014-01-05]. URL http://www.cisco.com/en/US/tech/tk365/technologies_tech_note09186a008010f016.shtml

[13] Cisco System, Inc.: Multi-Topology Routing [online]. Červenec 2010 [cit. 2014-02-05]. URL http://www.cisco.com/c/en/us/td/docs/ios/mtr/configuration/guide/12_2sr/mtr_12_sr_book/multi-top_rtng.html

[14] Cisco Systems, Inc.: Enhanced Interior Gateway Routing Protocol (EIGRP) Wide Metrics [Online]. Listopad 2012 [cit. 2014-01-05]. URL http://www.cisco.com/en/US/prod/collateral/iosswrel/ps6537/ps6554/ps6599/ps6630/whitepaper_C11-720525.html

[15] Cisco Systems, Inc.: An Introduction to IGRP [Online]. Srpen 2005 [cit. 2014-01-03]. URL http://www.cisco.com/en/US/tech/tk365/technologies_white_paper09186a00800c8ae1.shtml

[16] INET Framework: INET Framework for OMNeT++/OMNEST [Online]. [cit. 2014-01-03]. URL http://inet.omnetpp.org/doc/INET/neddoc/index.html

[17] INET Framework: *INET Framework for OMNeT++, Manual*. Červen 2012 [cit. 2014-01-03]. URL http://inet.omnetpp.org/doc/INET/inet-manual-draft.pdf

[18] J. J. Garcia-Lunes-Aceves: A Unified Approach to Loop-Free Routing using Distance Vectors or Link States. *ACM 089791-332-9/89/0009/0212*, 1989: s. 212–223.

[19] J. J. Garcia-Lunes-Aceves: Loop-Free Routing Using Diffusing Computations. *IEEE/ACM Transactions on Networking*, ročník Vol. 1, No. 1, Únor 1993: s. 130–141.

[20] Odom, W.: *CCNP ROUTE 642-902, Official Certification Guide*. Cisco Press, Leden 2010, iSBN: 978-1-58720-253-7.

[21] Savage, D.; Slice, D.: Enhanced Interior Gateway Routing Protocol [Online]. Únor 2013 [cit. 2014-01-03]. URL http://tools.ietf.org/html/draft-savage-eigrp-00

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

[23] Scott, J.: CISCO Router Load-Balancing Discussion. 2005 [cit. 2014-01-06]. URL http://tjscott.net/tcpip/eigrp.lb.pdf

[24] Teare, D.: *Implementing Cisco IP Routing (ROUTE).* Cisco Press, 2010, iSBN-13: 978-1-58705-882-0.

[25] Tlolka, M.: *Simulace EIGRP protokolu v prostředí OMNeT++.* Bakalářská práce, FIT VUT, Brno, 2009.

[26] Varga, A.: *OMNeT++, User Guide, Version 4.3.* 2011 [cit. 2014-01-05]. URL http://www.omnetpp.org/doc/omnetpp/manual/usman.html

Příloha A

Obsah CD

V tabulce A.1 je popsán obsah adresářů na přiloženém CD.

Adresář	
`/doc`	Vygenerovaná programová dokuentace ve formátu HTML.
`/examples`	Simulační příklady, na kterých byly provedeny testy v kapitole 7. Jedná se o soubory `basic` a `square`, jsou tu ale i další příklady.
`/rules`	Popis pravidel, která byla vypozorována na základě ladících výpisů a odchycené komunikace v simulátoru GNS3. Použitý IOS je 15.2(4) M2 a Cisco směrovače 7200 (c7200-adventerprisek9-mz.152-4.M2.bin).
`/src-eigrp`	Zdrojové soubory modulu EIGRP.
`/src-ansainet`	Zdrojové knihovny ANSAINET spolu s EIGRP modulem.
`/textsrc`	Zdrojové soubory tohoto textu pro systém LaTeX.
`/test`	Výstupy testů na rélné síti z kapitoly 7.

Tabulka A.1: Obsah CD

Příloha B

Verze EIGRP

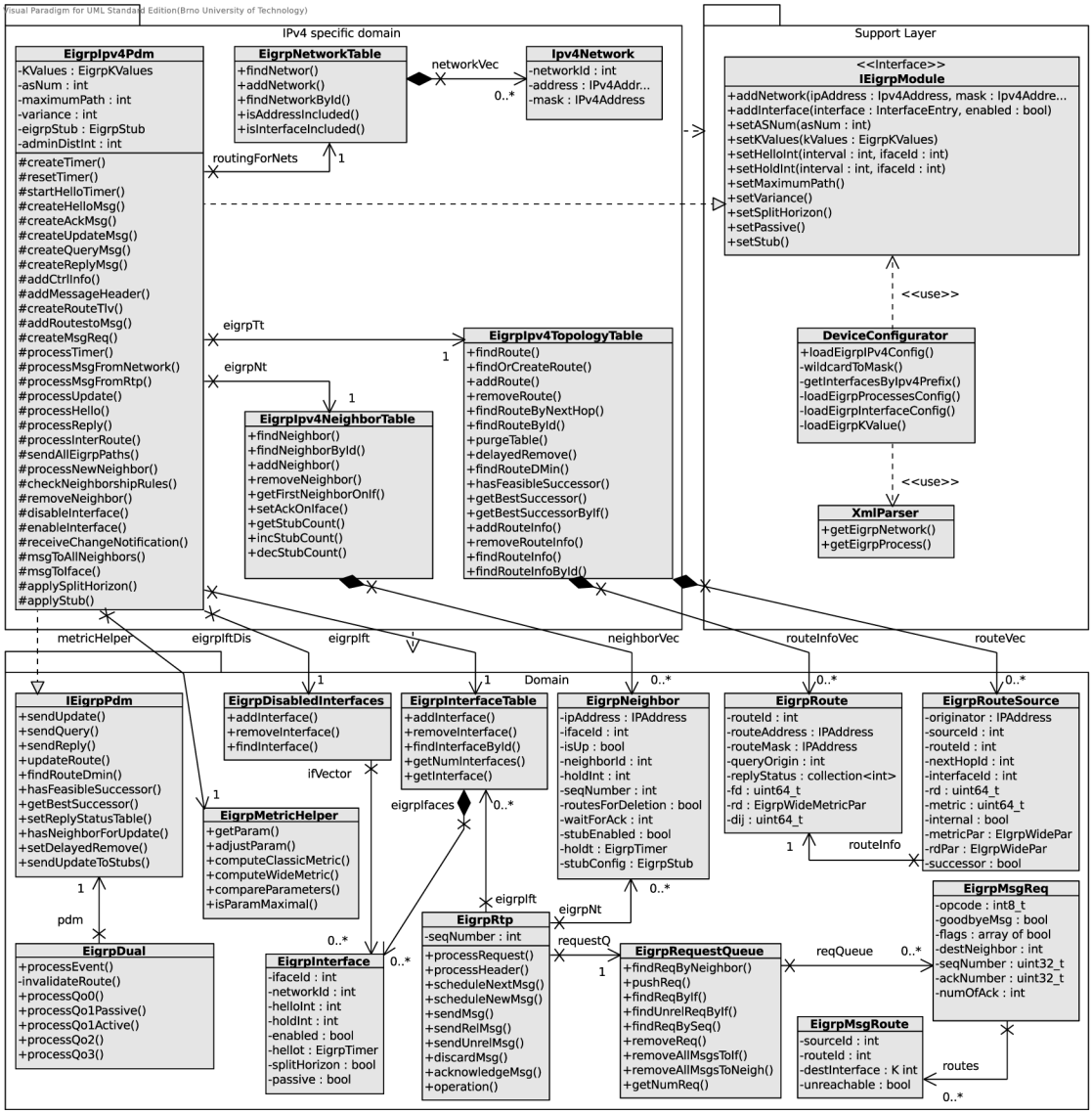
Níže je uveden soupis verzí EIGRP a operačních systémů IOS, jenž se mi podařilo zjistit. Některé verze EIGRP nejsou dostupné na určitých typech zařízení. Čerpal jsem především z Cisco matériálů vztahujících se ke konkrétním zařízením a z internetových diskuzí, jelikož se mi nepodařilo najít přehled verzí tohoto protokolu. Přehled je jen informativní.

Verze IOS	**Verze EIGRP**
10.3(11), 11.0(8), 11.1(3) a dřívější	EIGRP 0 [3]
10.3(11), 11.0(8), 11.1(3) a pozdější	EIGRP 1 [3]
12.2(50) a pozdější	EIGRP 6
15.1(3)	EIGRP 8
15.2(4)	EIGRP 10

Tabulka B.1: Verze EIGRP

Příloha C

Diagram tříd



Obrázek C.1: Diagram tříd