ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

KATEDRA INFORMAČNÝCH SIETÍ

RIEŠENIE SUSEDSKÝCH VZŤAHOV INTEGROVANÝCH SMEROVACÍCH PROTOKOLOV

Dizertačná práca

Ing. Martin Kontšek

Registračné číslo: 28360020203005

Žilina, 2020

ŽILINSKÁ UNIVERZITA V ŽILINE

FAKULTA RIADENIA A INFORMATIKY

KATEDRA INFORMAČNÝCH SIETÍ

RIEŠENIE SUSEDSKÝCH VZŤAHOV INTEGROVANÝCH SMEROVACÍCH PROTOKOLOV

Dizertačná práca

Študijný program: aplikovaná informatika
Študijný odbor: informatika
Pracovisko: Fakulta riadenia a informatiky, KIS
Vedúci práce: doc. Ing. Pavel Segeč, PhD.
Školiteľ špecialista: Ing. Peter Palúch, PhD.
Registračné číslo: 28360020203005

Žilina, 2020 **Ing. Martin Kontšek**

Čestné vyhlásenie

Čestne prehlasujem, že som prácu vypracoval samostatne s využitím dostupnej literatúry a vlastných vedomostí pod odborným vedením vedúceho práce doc. Ing. Pavla Segeča, PhD. a školiteľa špecialistu Ing. Petra Palúcha, PhD. Všetky zdroje použité v práci som uviedol v súlade s predpismi.

V Žiline, dňa 10.5.2020 Ing. Martin Kontšek

Poďakovanie

Moje poďakovanie patrí vedúcemu práce doc. Ing. Pavlovi Segečovi, PhD. a školiteľovi špecialistovi Ing. Petrovi Palúchovi, PhD., ako aj ďalším kolegom, za odbornú pomoc, pripomienky a usmerňovanie pri tvorbe práce. Taktiež by som chcel poďakovať mojej priateľke a rodine za neustálu podporu a povzbudzovanie, bez ktorých by táto práca nevznikla.

ABSTRAKT V ŠTÁTNOM JAZYKU

Kontšek, Martin: *Riešenie susedských vzťahov integrovaných smerovacích protokolov*. [Dizertačná práca] – Žilinská univerzita v Žiline, Fakulta riadenia a informatiky, Katedra informačných sietí. – Vedúci: doc. Ing. Pavel Segeč, PhD. – Školiteľ špecialista: Ing. Peter Palúch, PhD. – stupeň odbornej kvalifikácie: Doktor v odbore Aplikovaná informatika. Žilina: FRI ŽU v Žiline, 2020. – 107 s.

Cieľom predloženej diplomovej práce je vytvoriť odporúčania pre dizajn integrovaných smerovacích protokolov. Prvá časť práce popisuje súčasné integrované smerovacie protokoly, ich históriu a kľúčové vlastnosti. Následne popisuje identifikované triedy riešenia protokolovej integrovanosti na základe počtu susedstiev a protokolu využitého na transport smerovacích správ. V ďalšej časti sú popísané formálne metódy, z ktorých boli farbené Petriho siete vybraté ako prostriedok na vytvorenie modelov jednotlivých identifikovaných tried. Časť práce je venovaná detailnej analýze protokolu EIGRP, ktorý bol vybratý ako vzor pre modelovanie tretej triedy. V ďalších častiach práce sú detailne popísané a analyzované jednotlivé vytvorené modely tried riešenia protokolovej integrovanosti, následne verifikované a porovnané. Na základe porovnania modelov boli v poslednej časti práce formulované odporúčania pre dizajn integrovaných smerovacích protokolov.

Kľúčové slová: odporúčania, smerovací protokol, susedské relácie, IPv4, IPv6

ABSTRAKT V CUDZOM JAZYKU

Kontšek, Martin: *Neighbor Session Solutions for Integrated Routing Protocols*. [Dissertation thesis] – University of Žilina, Faculty of Management Science and Informatics, Department of InfoComm Networks. – Tutor: doc. Ing. Pavel Segeč, PhD. – Specialist tutor: Ing. Peter Palúch, PhD. – Qualification level: Doctor of Applied Informatics. Žilina: FRI ŽU in Žilina, 2020. – 107 p.

The aim of this dissertation thesis is to propose a methodology for design of integrated routing protocols. First part of the thesis describes currently used integrated routing protocols, their history and key features. The description of four identified classes of protocol integration based on number of neighbor sessions and protocol used for transport of routing messages is presented afterwards. Next part deals with summary of formal methods, one of which was selected as a tool to create a model of each protocol integration class. The selected formal method is named Colored Petri Nets. One part of the publication is devoted to detail analysis of EIGRP routing protocol, which was chosen as a base for model of the third protocol integration class. Following parts deal with detail description and analysis of created protocol integration class models, which are also verified and compared with each other. Based on the comparison of the each integration class model, last part formulates recommendations for integrated routing protocol design.

Key words: recommendations, routing protocol, neighbor sessions, IPv4, IPv6

Obsah

Zoznam obrázkov

Zoznam tabuliek

Zoznam skratiek

ABR	Area Border Router
AS	Autonomous System
BGP	Border Gateway Protocol
CD	Computed Distance
CLI	Command-line Interface
CPN	Colored Petri Nets
CPN ML	Colored Petri Nets Meta Language
CR	Conditional Receive
CSNP	Complete Sequence Number PDU
DUAL	Diffusing Update Algorithm
EGP	Exterior Gateway Protocols
EIGRP	Enhanced Interior Gateway Routing Protocol
EOT	End-of-Table
FC	Feasibility Condition
FD	Feasible Distance
FDT	Formal Description Techniques
FRR	Free Range Routing
FSM	Finite State Machine
GNS3	Graphical Network Simulator-3
GPL	GNU General Public License
IANA	Internet Assigned Numbers Authority
IDE	Integrated Development Environment
IGP	Interior Gateway Protocols
IGRP	Interior Gateway Routing Protocol
IOS	Internetwork Operating System
IP	Internet Protocol
IS-IS	Intermediate System to Intermediate System
ISP	Internet Service Provider
LSA	Link-State Advertisement
LSP	Link-State PDU

MP-BGP	Multiprotocol BGP
MPLS	Multiprotocol Label Swtching
MTU	Maximum Transmission Unit
NSAP	Network Service Access Point
OSI	Open Systems Interconnection
OSPF	Open Shortest Path First
PDU	Protocol Data Unit
PIM	Protocol Independent Multicast
PSNP	Partial Sequence Number PDU
RD	Reported Distance
RIP	Routing Information Protocol
RIPng	RIP Next Generation
RTP	Reliable Transport Protocol
SIA	Stuck in Active
SNMP	Simple Network Management Protocol
SML	Standard Meta Language
TLV	Type-Length-Value

Úvod

Neodmysliteľnou súčasťou nášho každodenného života sú počítače, smartfóny a aj iné zariadenia, pomocou ktorých vykonávame svoju prácu, nakupujeme a komunikujeme s priateľmi, ktorí sa nachádzajú na rôznych miestach planéty. Tieto zariadenia nám taktiež sprostredkúvajú rôzne iné služby, ktoré boli v minulosti nemysliteľné, ako napríklad narábanie s účtom v banke, nákup cestovného lístka na autobus alebo vlak online, alebo aj prístup k množstvu informácií odkiaľkoľvek. Napriek tomu si množstvo ľudí neuvedomuje, že tieto služby by neboli možné bez počítačových sietí, ako sú Internet, ale aj lokálne počítačové siete vo firmách, školách aj našich domácnostiach.

Hlavným prvkom počítačovej siete je smerovač, ktorý prepája viaceré siete a je zodpovedný za prenos paketov medzi nimi na základe cieľovej IP adresy, ktorá môže prislúchať zariadeniu, ako je napríklad webový server v korporátnej sieti, alebo emailový server v zahraničí. Keďže takmer v každej lokalite, okrem domácich sietí alebo malých firiem, sa nachádza viacero smerovačov, je potrebné, aby každý z nich mal informácie o trasách do konkrétnych cieľových sietí. Tieto informácie je možné nakonfigurovať manuálne, čo je vhodné iba pri malom počte smerovačov, alebo je možné použiť jeden zo smerovacích protokolov, pomocou ktorých si smerovače vymenia smerovacie informácie automaticky.

Aj napriek tomu, že smerovacie protokoly sa vyvíjajú už dlhú dobu, ich vývoj nie je ani zďaleka ukončený, pretože prichádzajú nové trendy, ako napríklad SDN (softvérovo definované siete), používajú sa viaceré adresové rodiny súčasne (IPv4 a IPv6) a smerovacie protokoly sa začínajú objavovať aj v iných oblastiach, ako napríklad MANET (mobilná ad-hoc sieť) a VANET (automobilová ad-hoc sieť) siete.

Práve prostredie IP sietí, v ktorých sa používa „*starý*“ IPv4 protokol a zvažuje sa, alebo dokonca už je tiež prevádzkovaný aj „*nový*“ IPv6 protokol, stále čelí rôznym problémom alebo prekážkam. Jednou z problematických oblastí je oblasť nasadenia a prevádzkovania smerovacích protokolov v multiprotokolovom prostredí (IPv4 a IPv6 protokol súčasne). Štúdiom dostupnej literatúry a experimentmi bolo zistené, že každý v súčasnosti dostupný smerovací protokol rieši problém svojej činnosti a budovania IPv4/IPv6 smerovacích informácii iným, svojím spôsobom. Vybrané smerovacie protokoly používané v multiprotokolovom prostredí IP sietí sú preto popísané aj spolu s princípmi smerovania v IP sieťach v kapitole 1.

Keďže problematika integrovaných smerovacích protokolov, teda protokolov podporujúcich IPv4 aj IPv6 súčasne, je veľmi široká, počas riešenia dizertačnej práce padlo rozhodnutie na zúženie oblasti skúmania na problematiku vytvárania susedských vzťahov a protokolov využitých na prenos správ smerovacieho protokolu.

Následne sa nám počas analýzy existujúcich integrovaných protokolov podarilo identifikovať štyri kombinácie počtu vytváraných susedstiev a protokolu slúžiaceho na prenos správ samotného smerovacieho protokolu. Tieto kombinácie, ktoré sú v práci nazývané ako triedy riešenia protokolovej integrovanosti, sú spolu so službami nevyhnutnými na vytvorenie susedstiev a prenos informácii popísané v kapitole 2.

Cieľom práce bolo objasniť, ktorá z tried riešenia protokolovej integrovanosti je efektívnejšia a následne navrhnúť všeobecné odporúčania využiteľné pre budúci dizajn a implementáciu smerovacích protokolov, ktorý je vhodný práve pre nasadenia v prostredí sietí využívajúcich viaceré sieťové protokoly a adresové systémy. Aby bolo možné dosiahnuť tento cieľ, vykonali sme analýzu formálnych metód, popísanú v kapitole 3, z ktorej vyplynulo, že jednou z najvhodnejších metód je použitie farbených Petriho sietí.

Naša katedra má so smerovacími protokolmi skúsenosti aj vďaka vývoju otvorenej implementácie smerovacieho protokolu EIGRP. Preto padlo rozhodnutie na začatie modelovania treťou triedou riešenia protokolovej integrovanosti, keďže smerovací protokol EIGRP patrí práve do tejto triedy. Nadobudnuté znalosti z hlbšej analýzy dizajnu protokolu, popísané v kapitole 4, sme pretavili práve do konceptuálneho modelu tretej triedy riešenia protokolovej integrovanosti, od ktorej sme odvádzali modely ostatných tried.

Vytvorené modely jednotlivých tried boli do detailov popísané, ich dizajn verifikovaný a následne porovnaný medzi sebou na základe vybraných kritérií. Na základe výsledkov porovnania bolo neskôr možné stanoviť odporúčania pre dizajn budúcich, ale aj úpravu súčasných integrovaných smerovacích protokolov.

1 Smerovanie v sieťach

Ako už bolo spomenuté v úvode, aby mohli zariadenia v našej sieti komunikovať so zariadeniami nachádzajúcimi sa vo vzdialenej sieti, je potrebné, aby bol v našej sieti smerovač. Tento prepája našu sieť s inou sieťou (či sieťami) a má informácie, ako dosiahnuť túto vzdialenú cieľovú sieť (tzv. smerovacie informácie). Smerovač má tieto informácie uložené vo forme položiek v smerovacej tabuľke a aktuálnosť týchto informácií zabezpečuje efektívne smerovanie paketov prechádzajúcich cez smerovač [1], [2].

Keď smerovač prijme IP paket, na základe cieľovej adresy zapísanej v hlavičke paketu prehľadá svoju smerovaciu tabuľku, kde sa pokúsi nájsť najlepšiu cestu do siete, ktorá prislúcha cieľovej adrese. Ak sa položka v smerovacej tabuľke nájde, smerovač zabalí paket do rámca a pošle ho susednému smerovaču cez výstupné rozhranie prislúchajúce nájdenému záznamu. Ak sa prislúchajúci záznam nenájde a k dispozícii je predvolená cesta (angl. *default route*), smerovač pošle paket cez rozhranie prislúchajúce tejto default route. Ak sa ale nenájde ani default route, smerovač paket zahodí [3], [4].

Do smerovacej tabuľky sa automaticky pridávajú záznamy o sieťach priamo pripojených k smerovaču. Ostatné smerovacie záznamy o vzdialených sieťach je potrebné pridať buď manuálnou konfiguráciou, alebo použiť dynamický smerovací protokol. V prípade podpory viacerých sieťových protokolov má smerovač pre každý z nich vytvorenú samostatnú smerovaciu tabuľku [3]–[5].

1.1 Statické smerovacie záznamy

Statické smerovacie záznamy priamo určujú cestu do cieľovej siete určením výstupného rozhrania alebo nasledujúceho susedného smerovača na ceste do cieľa. Výhodou statických záznamov, na rozdiel od dynamických smerovacích protokolov, je zvýšená bezpečnosť a nenáročnosť na prostriedky smerovača, keďže smerovač nemusí prijímať a spracovávať správy o cestách od susedov a z prijatých informácií vypočítavať najkratšie cesty do cieľových sietí. Zvýšená bezpečnosť vyplýva z manuálnej konfigurácie administrátorom, pretože v prípade použitia chýbajúcej alebo nedostatočnej konfigurácie zabezpečenia dynamického smerovacieho protokolu, môže útočník podvrhnúť smerovaču cestu do vzdialenej siete, a tým presmerovať požadovanú komunikáciu sieťových zariadení [1], [2], [4].

V prípade, že sa sieťová topológia zmení, je potrebné zmeniť smerovací záznam manuálne, alebo pridať nový záznam, čo je spolu so vzrastajúcou administratívnou náročnosťou v závislosti od počtu ciest hlavná nevýhoda statického smerovania. Ďalšou z nevýhod statických smerovacích záznamov je aj ich náchylnosť na vznik chýb spôsobených ako následok preklepov a nepresností pri zadávaní administrátorom, hlavne v prípade spomínaného zadávania väčšieho množstva statických smerovacích záznamov. Preto sa v rozľahlejších sieťach využívajú prevažne dynamické smerovacie protokoly [4].

1.2 Dynamické smerovacie protokoly

Účelom dynamických smerovacích protokolov je automatické zisťovanie informácií o dostupnosti a stave vzdialených sietí, výmena informácií so susednými smerovačmi a napĺňanie smerovacích tabuliek. Dôležitou vlastnosťou smerovacích protokolov je čo najefektívnejšia výmena smerovacích informácií medzi susednými smerovačmi, aby sa neobjavovali smerovacie slučky, následkom ktorých sa narúša prenos dát a stabilita siete [1], [2].

Dynamické smerovacie protokoly sa rozdeľujú podľa rôznych kritérií, ako napríklad podľa oblasti použitia alebo typu vymieňaných smerovacích informácií posielaných medzi susedmi. Ako hlavné delenie sa v literatúre uvádza delenie podľa toho, či sa smerovací protokol používa v rámci autonómneho systému (AS) alebo medzi rôznymi autonómnymi systémami. Pod pojmom autonómny systém je typicky chápaná sieť alebo jej časť, ktorá je pod kontrolou jednej entity. Veľké spoločnosti sú spravidla reprezentované jedným AS a ak máme napríklad pripojenie ku dvom rôznym poskytovateľom sieťovej konektivity (ISP), každý z nich má vlastný autonómny systém. Každý autonómny systém je rozlíšený unikátnym číslom, pričom tieto čísla sú prideľované organizáciou IANA [4].

Sieťový protokol IPv6 prišiel neskôr ako IPv4 a v prípade väčšiny vtedajších smerovacích protokolov ich dizajnéri s podporou viacerých sieťových protokolov nerátali. Preto niektoré protokoly, ako napríklad RIPv2 alebo OSPF sieťový protokol nepodporujú vôbec a bola vytvorená nová verzia, s podporou výlučne IPv6 protokolu. Iné protokoly boli o podporu IPv6 rozšírené, čo je prípad napríklad protokolu IS-IS [4], [5].

1.3 Vonkajšie smerovacie protokoly

Smerovacie protokoly používané medzi autonómnymi systémami sa nazývajú vonkajšie smerovacie protokoly (angl. *Exterior Gateway Protocols - EGP*). V minulosti sa využíval protokol Exterior Gateway Protocol, ktorý bol prvým smerovacím protokolom, keďže vznikol už v roku 1982. V súčasnosti do tejto kategórie spadá jediný protokol s názvom Border Gateway Protocol (BGP) [2].

BGP je takzvaný path-vector smerovací protokol, čo znamená, že si so susedmi vymieňa informácie o ceste do konkrétnej cieľovej siete a jej atribútoch. Tieto informácie o presnej trase sú reprezentované postupnosťou čísel autonómnych systémov. Pôvodná verzia protokolu BGP podporovala len IPv4 adresy. Súčasná verzia BGP-4 s multiprotokolovým rozšírením (MP-BGP) podporuje smerovanie oboch verzií IP protokolu (IPv4 aj IPv6) i prenos iných informácií v správach, ako napr. MPLS návestí. Protokol BGP prenáša správy pomocou transportného protokolu TCP. BGP, na rozdiel od iných smerovacích protokolov, umožňuje prenášať smerovacie informácie nezávisle od použitého sieťového protokolu na transport správ. Preto je možné napríklad pomocou IPv4 susedstva prenášať IPv4 aj IPv6 smerovacie informácie. Vďaka faktu, že BGP je v súčasnosti jediný používaný EGP protokol, považuje sa za smerovací protokol, ktorý „poháňa Internet“. BGP je veľmi dobre škálovateľný, keďže v súčasnosti prenáša viac ako 700 tisíc prefixov [6]. Zároveň je BGP stabilný a jeho pomalá konvergencia v prípade topologických zmien je v tomto prípade výhodou, keďže prispieva k stabilite smerovacích informácií a redukuje následky zlyhaní smerovačov, ako je napríklad port neustále meniaci svoj stav (angl. *port flapping*). Protokol BGP je možné v niektorých prípadoch, napríklad v spolupráci s OSPF, použiť aj ako vnútorný smerovací protokol [4], [7].

1.4 Vnútorné smerovacie protokoly

Vnútorné smerovacie protokoly (angl. *Interior Gateway Protocols - IGP*) je názov skupiny smerovacích protokolov využívaných v rámci jedného autonómneho systému. Vnútorných smerovacích protokolov, na rozdiel od vonkajších, sa používa viacero, a taktiež sú na ne kladené iné požiadavky, čo vyplýva z použitia v inej oblasti. Najčastejším delením vnútorných smerovacích protokolov je delenie podľa typu štruktúr, pomocou ktorých si smerovače vymieňajú informácie na: distance-vector a link-state [4].

Susedné smerovače si prostredníctvom distance-vector smerovacích protokolov vymieňajú informácie o známych sieťach vo forme jednorozmerných polí (vektorov) vzdialeností. Vzdialenosť, v tomto prípade, reprezentuje mieru dostupnosti, alebo metriku konkrétnej cieľovej siete. Susedný smerovač, od ktorého sa dozvieme o existencii konkrétnej cieľovej siete, sa stane nasledujúcim (next hop) smerovačom na ceste do cieľovej siete. Ak nám rovnakú cieľovú sieť oznámilo viacero susedov, ako next hop bude vybratý ten, cesta cez ktorého bude mať najnižšiu celkovú metriku. Smerovač následne vybratú cestu pridá do smerovacej tabuľky, a následne svojim susedom ohlási svoje záznamy, ktoré sú aktuálne dostupné v jeho smerovacej tabuľke spolu s vlastnou vzdialenosťou do cieľovej siete. Z popísaného princípu fungovania distance-vector smerovacích protokolov vyplýva, že zo smerovacích informácií, ktoré majú k dispozícii, nie je možné detailne zrekonštruovať sieťovú topológiu. Spracovanie týchto informácií je síce na pamäť a výpočtový výkon smerovača nenáročné, ale jednoduchosť a neúplnosť týchto informácií ako aj spôsob výberu cesty distance-vector protokolmi môže mať za následok náchylnosť na vznik smerovacích slučiek. Medzi v súčasnosti používané distance-vector smerovacie protokoly patria RIPv2, RIPng a EIGRP [3].

Na rozdiel od distance-vector protokolov si link-state protokoly vymieňajú informácie o individuálnych objektoch sieťovej topológie a spôsobe ich prepojenia. Medzi tieto objekty patria smerovače buď vo vnútri autonómneho systému, na hranici s iným AS, alebo siete z iných AS alebo oblastí. Po spustení link-state smerovacieho protokolu vytvorí smerovač správu, v ktorej popíše seba a linky vedúce k priamym susedom. Táto správa je bez zmien záplavovo šírená po celej oblasti, a tak ju dostane každý smerovač s aktívnym link-state smerovacím protokolom. Preto má každý smerovač informácie o celej sieťovej topológii, pozná každý smerovač, každú sieť, ako aj každú linku medzi susednými smerovačmi. Vďaka týmto detailným informáciám môže každý smerovač vytvoriť orientovaný graf topológie a v ňom vypočítať cesty s najmenšou metrikou do cieľových sietí. Preto sú link-state smerovacie protokoly oveľa menej náchylné na vznik smerovacích slučiek, ale aj nároky na pamäť a výpočtové prostriedky smerovača sú vyššie. Medzi súčasné link-state protokoly patria OSPF a IS-IS [3], [4].

1.4.1 Routing Information Protocol

Jedným z najstarších dynamických smerovacích protokolov je RIP, ktorého prvá verzia bola uvoľnená už v roku 1982 a neskôr v roku 1988 aj štandardizovaná. Ako už bolo spomenuté, patrí medzi distance-vector protokoly. Protokol RIP bol navrhnutý pre prostredie

malých sietí pracujúcich s protokolom IPv4, keďže podporuje len malú sieťovú topológiu (najviac 15 smerovačov v rade), a taktiež čas konvergencie siete pri zmene v topológii je relatívne dlhý. Ako všetky smerovacie protokoly tejto doby, aj RIPv1 je tzv. „classful" smerovací protokol, čo znamená, že smerovacie informácie neobsahujú sieťovú masku, ktorá je ale určená podľa toho, do ktorej triedy konkrétna sieť patrí, alebo podľa masky rozhrania, ktorým bola prijatá smerovacia informácia. Ako transportný protokol sa na prenos správ využíva UDP s číslom portu 520. RIP, na rozdiel od iných smerovacích protokolov, nevytvára susedstvá, ale pravidelne (štandardne každých 30 sekúnd) pošle všetky smerovacie informácie cez všetky rozhrania, na ktorých je protokol aktivovaný ako broadcast [2], [4].

Druhá verzia protokolu, ktorá bola vydaná v roku 1994 odstraňuje niektoré nedostatky. Bola pridaná podpora variabilných sieťových masiek (maska bola do smerovacích informácií pridaná) a manuálnej sumarizácie. Na rozdiel od prvej verzie, RIPv2 neposiela smerovacie informácie ako broadcast, ale aby stanice, ktoré nemajú smerovací protokol nakonfigurovaný správy nedostávali, používa sa multicast na adresu 224.0.0.9. Neskôr v roku 1997 bola pridaná aj podpora autentifikácie, ale podporovaná veľkosť topológie sa nezmenila. Oba protokoly (RIPv1 aj RIPv2) podporujú len smerovacie informácie s IPv4 adresami [2], [4].

Keďže smerovací protokol RIP bol vytvorený ešte pred nástupom IPv6 protokolu, jeho dizajnéri nepočítali s podporou iného sieťového protokolu ako IPv4. Keďže monolitická architektúra RIP-u neumožňuje jeho ľahké rozšírenie o iné sieťové protokoly, bolo ho potrebné úplne prepracovať. Podporu IPv6 sietí priniesla až nová verzia RIP protokolu s názvom Routing Information Protocol Next Generation (RIPng), ktorý bol štandardizovaný v roku 1997. S jeho predchodcom RIPv2 zdieľa všetky vlastnosti (metrika, podpora variabilných masiek, malá sieťová topológia), iba správy posiela na multicast adresu FF02::9 a rovnako ako predchádzajúce verzie využíva transportný protokol UDP ale s číslom portu 521. RIPng je avšak možné použiť iba na prenos IPv6 smerovacích informácií. Keďže podpora IPv6 nebola do predchádzajúcich verzií RIP protokolu pridaná, je v multiprotokolovom prostredí potrebné nasadiť RIPv2 pre IPv4 protokol a RIPng pre IPv6 protokol [3], [4].

1.4.2 Open Shortest Path First

Open Shortest Path First (OSPF) je zástupcom smerovacích protokolov typu link-state, jeho vývoj začal v roku 1987 a prvá verzia bola štandardizovaná v roku 1989.

Štandard OSPFv2, ktorý sa používa aj v súčasnosti, bol vydaný v roku 1998. OSPFv2 podporuje len protokol IPv4, pričom hlavičky správ sú priamo vložené v IPv4 protokole [2].

Po aktivácií OSPFv2 procesu na rozhraní sa začne vyhľadávanie susedov. Smerovač posiela a prijíma Hello správy na multicast adrese 224.0.0.5 a ak príjme Hello paket od iného smerovača overí si zhodu niektorých parametrov (autentifikácia, časovače, číslo oblasti). Po vytvorení susedskej relácie sa Hello pakety posielajú ako overenie existencie suseda. Ak sú smerovače v sieti s mnohonásobným prístupom (angl. *multiaccess*), prebehne voľba designated router (DR) a backup designated router (BDR) smerovača. Ak sa v sieti DR nachádza, smerovače synchronizujú databázy len priamo s DR, nie s každým smerovačom na multiaccess sieti. Následne si smerovače vymenia pakety obsahujúce hlavičky LSA paketov, na základe ktorých potom žiadajú od suseda alebo DR konkrétne LSA, ktoré smerovač v databáze nemá, alebo má staršiu verziu. LSA pakety sa používajú na prenos informácií o topológií siete. Akonáhle si smerovače zosynchronizujú pracovné databázy (mali by byť po synchronizácii zhodné), každý smerovač nezávisle spustí Dijikstrov algoritmus na vyhľadanie najkratších ciest (angl. *Shortest Path First - SPF*). Vybraté cesty sú následne vložené do smerovacej tabuľky. OSPF používa ako metriku ohodnotenie (angl. *cost*) rozhrania. OPSF je vhodný na použitie v rozľahlých sieťach a poskytuje rýchlu konvergenciu v prípade topologických zmien v sieti [3], [4].

Protokol OSPF člení sieť do oblastí, pričom každá sieť musí obsahovať aspoň jednu oblasť (tzv. backbone area 0). Hraničné smerovače, ktoré sú na rozhraní medzi oblasťami, sa nazývajú Area Border Router (ABR). Každý ABR smerovač musí byť pripojený do backbone oblasti. Členenie siete na oblasti ovplyvňuje aj typy posielaných LSA štruktúr [3].

Keďže podobne ako pri protokole RIP sa pri návrhu OSPF nepočítalo s podporou viacerých sieťových protokolov, ani architektúra OSPF neumožňuje jednoduché rozšírenie. Preto podpora pre IPv6 siete bola pridaná až do novej verzie protokolu OSPFv3, ktorého prvá špecifikácia bola uvoľnená v roku 1999. Používa rovnaký princíp vytvárania susedstiev, výmeny topologických informácií a výpočtu najkratších ciest ako jeho predchodca, ale využíva aj iné typy LSA štruktúr a niektoré nazýva inak. Na prenos informácií využíva IPv6 protokol, do ktorého priamo vkladá hlavičky svojich paketov.

Pôvodný štandard OSPFv3 podporuje iba IPv6 protokol, avšak v roku 2010 bolo vydané rozšírenie RFC 5838 [8], ktoré do OSPFv3 pridalo podporu prenosu IPv4 smerovacích informácií. V rámci jedného procesu je vďaka tomu možné prenášať

smerovacie informácie o IPv4 aj IPv6, avšak pre každý protokol je potrebné nadviazať susedstvo zvlášť. Transport je v oboch prípadoch zabezpečený pomocou IPv6 protokolu, pričom pakety prislúchajúce jednotlivým susedstvám sú rozlíšené v hlavičke OSPF paketu poľom s názvom ID inštancie (angl. *Instance ID*) [4].

1.4.3 Intermediate System to Intermediate System

Intermediate System to Intermediate System (IS-IS) je rovnako ako OSPF link-state smerovací protokol, ktorý je vhodný na použitie v rozľahlých sieťach a poskytuje veľmi rýchlu konvergenciu v prípade topologických zmien. Podobne ako protokol OSPF používa na výpočet najkratších ciest Dijikstrov SPF algoritmus. Pôvodne bol vyvinutý pre OSI siete ako štandard ISO-IEC 10589:2002, avšak nie je silne závislý od OSI protokolov, a preto ho bolo možné použiť s drobnými modifikáciami aj pre IP siete. Táto špecifikácia bola zverejnená ako RFC 1195 a modifikovaný protokol sa nazýva aj ako integrovaný IS-IS [3].

Adresovanie v IS-IS je riešené pomocou OSI adresy s názvom Network Service Access Point (NSAP) a na rozdiel od IP adries sa používa iba jedna adresa pre celé zariadenie, nie pre každé sieťové rozhranie [3], [9].

IS-IS ako jediný smerovací protokol neprenáša správy pomocou žiadneho sieťového protokolu (L3 vrstva ISO OSI), ale správy vkladá priamo do linkových rámcov. Informácie o adresovaní a susedstvách prenáša v štruktúrach pozostávajúcich z typu, dĺžky štruktúry a samostatného obsahu, preto sa zvyknú označovať ako TLV štruktúry (angl. *Type-Length-Value - TLV*). Vďaka TLV štruktúram je protokol možné jednoducho rozšíriť a disponuje väčšou flexibilitou. Preto podporuje viaceré sieťové protokoly (označuje sa aj ako protokolovo agnostický) a na rozdiel od iných smerovacích protokolov umožňuje vypočítať najkratšie cesty pre všetky sieťové protokoly spoločne, ale aj oddelene. Vďaka použitému dizajnu bola v roku 2008 do IS-IS pridaná aj podpora pre IPv6, popísaná v RFC 5308. Medzi jeho ďalšie funkcie patrí aj podpora variabilných sieťových masiek, ľubovoľná sumarizácia, rozdelenie autonómneho systému na oblasti i autentifikácia [3], [9].

IS-IS rozlišuje dve úrovne smerovania: Level1 a Level2. Smerovanie Level1 zabezpečuje smerovanie IP sietí len vo vnútri aktuálnej oblasti a smerovače majú k dispozícii informácie o topológii a najkratších cestách v celej oblasti. Level2 smerovače zabezpečujú smerovanie medzi oblasťami a majú informácie o konkrétnych, alebo sumárnych sieťach zo všetkých oblastí. Aby smerovače mohli vytvoriť susedstvo, musí sa úroveň smerovania zhodovať. V prípade, že smerovač podporuje oba levely, vytvárajú sa

samostatné susedstvá pre každý level. Chrbtica siete v IS-IS je tvorená súvislou postupnosťou smerovačov s úrovňou smerovania Level2, pričom v sieti môže byť len jedna [3], [9].

Identifikátory oblastí sú súčasťou NSAP adresy smerovača, z čoho vyplýva, že smerovač môže byť súčasťou len jednej oblasti, a teda hranice oblastí sú na rozdiel od OSPF na linkách medzi smerovačmi. Smerovače, ktoré podporujú obe úrovne smerovania, slúžia ako hraničné smerovače medzi Level1 a Level2 smerovaním. V niektorých prípadoch je ale možné, aby jeden smerovač patril do viacerých oblastí pomocou konfigurácie viacerých adries na jednom smerovači. Podľa odporúčaní by viaceré adresy nemali byť na smerovači nakonfigurované trvale, ale iba ako prostriedok na dosiahnutie určitých zmien v topológii bez prerušenia prevádzky siete, ako napríklad prečíslovanie oblastí [3].

Typy paketov používané smerovacím protokolom IS-IS sú štyri: Hello, Link-state PDU (LSP), Complete Sequence Number PDU (CSNP) a Partial Sequence Number PDU (PSNP).

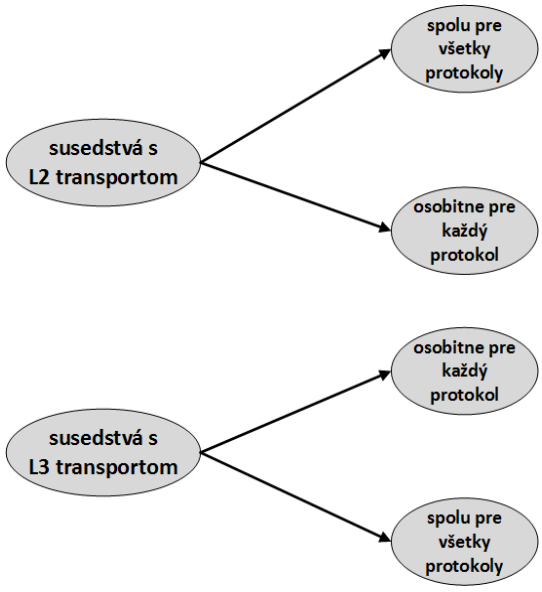
Hello pakety slúžia na detekciu smerovačov a vytváranie a udržiavanie susedských relácií. V prípade broadcast sietí sa posielajú samostatné L1 a L2 hello pakety podľa úrovne smerovania podporovanej smerovačom. V prípade point-to-point linky sa pre zvýšenie efektivity posiela jeden spoločný hello paket pre obe úrovne smerovania. Hello pakety sa štandardne posielajú každých 10 sekúnd. Hold time sa nastavuje multiplikátorom voči hello intervalu a štandardne je nastavený na 3-násobok. Na rozdiel od OSPF sa časovače medzi susedmi nemusia zhodovať [3].

LSP pakety slúžia na distribúciu smerovacích informácií medzi smerovačmi a podobajú sa na LSA v OSPF. LSP pakety sú číslované hodnotami od 0x00000001 po 0xFFFFFFFF. Ak číslo LSP nadobudne maximálnu hodnotu, musí smerovač počkať aspoň 21 minút, aby LSP exspirovalo z databáz ostatných smerovačov, a potom môže poslať LSP s počiatočnou hodnotou. Platnosť LSP je štandardne 20 minút, pričom sa zvyčajne obnovuje každých 15 minút. Keďže IS-IS správy sú zabalené priamo do linkových rámcov, ktorých MTU je limitované, musí IS-IS riešiť fragmentáciu správ. Riešenie je v tomto prípade jednoduché, ak je potrebné v LSP poslať veľký počet TLV, ktoré by presiahli MTU rámca, pošle sa viacero LSP správ. Pomocou CSNP paketu smerovač oznamuje susedom zoznam LSP, ktoré má k dispozícii vo svojej link-state databáze. Na vyžiadanie konkrétneho LSP a potvrdenie doručenia LSP slúžia PSNP pakety [3].

2 Aspekty dôležité pre podporu viacerých sieťových protokolov

Na základe dlhoročných skúseností členov Katedry informačných sietí v oblasti sieťových protokolov, ako aj súčasných trendov v oblasti integrovaných smerovacích protokolov sme pristúpili k zúženiu oblasti skúmania. Na základe analýzy dostupnej odbornej literatúry sme sa zamerali na podporu viacerých sieťových protokolov z pohľadu protokolu použitého na transport (prenos) smerovacích informácií a na otázku počtu susedských relácií špecificky pre oblasť, kedy sú v sieti nasadené viaceré sieťové protokoly, keďže táto problematika doteraz nebola riešená. V otázke použitého transportného protokolu nás zaujíma riešenie transportu jednak vzhľadom na vrstvu komunikačného modelu (L2 vs. L3), ako aj na problematiku prenosu samotnej smerovacej informácie. V oblasti susedstva nás podobne zaujímajú otázky riešenia susedských vzťahov opäť jednak na vrstvu komunikačného modelu, ako aj použité L3 protokoly.

Na základe analýzy sa nám podarilo identifikovať štyri možné triedy kombinácií spôsobu riešenia transportu smerovacích informácii a počtu susedských relácií vzhľadom na viaceré použité sieťové protokoly, ktoré sú znázornené na obrázku (Obrázok 1).



Obrázok 1: Triedy kombinácií transportu a počtu susedstiev, zdroj autor

Ako vyplýva z obrázka, možné kombinácie riešenia, nazvime ich ďalej triedy riešenia protokolovej integrovanosti, sú nasledovné:

1. Riešenie transportu smerovacích informácií na linkovej vrstve, riadenie susedstva osobitné pre každý sieťový protokol
2. transport na linkovej vrstve, jedno susedstvo pre všetky použité sieťové protokoly
3. transport na sieťovej vrstve, osobitné susedstvo pre každý sieťový protokol
4. transport na sieťovej vrstve, jedno susedstvo pre všetky použité sieťové protokoly.

Súčasné smerovacie protokoly podporujúce oba sieťové protokoly (IPv4 aj IPv6) môžeme rozdeliť nasledovne:

EIGRP, keďže využíva vlastný transportný protokol RTP a jeho pakety balí priamo do IP protokolu, pričom vytvára samostatné susedstvá pre každý sieťový protokol, zaraďujeme do triedy 3.

Smerovací protokol OSPFv3, ako bolo spomenuté v kapitole 1.4.2, vytvára samostatné susedstvá pre každý sieťový protokol a ich správy posiela v oboch prípadoch pomocou IPv6 sieťového protokolu. Preto je rovnako ako EIGRP zaradený do triedy 3.

IS-IS vďaka jeho dizajnu ešte pre OSI siete a jeho protokolovú nezávislosť, zaraďujeme do triedy 2, keďže na prenos správ používa priamo rámce linkovej vrstvy a v prípade prenosu smerovacích informácií IPv4 aj IPv6 protokolu vytvára iba jedno susedstvo.

Výnimku tvorí len smerovací protokol BGP s multiprotokolovým rozšírením, ktorý môžeme zaradiť do triedy 3 aj 4, keďže umožňuje v rámci jedného susedstva prenášať informácie IPv4 aj IPv6, ale je možné vytvoriť aj viaceré susedstvá a prenášať smerovacie informácie samostatne, ako bolo popísané v kapitole 1.3.

Pri riešení problematiky „integrovanosti“ a jej vplyvu na dizajn protokolu je potrebné sa zamyslieť nad oblasťami implementácie častí smerovacieho protokolu súvisiacimi s triedami riešenia integrovanosti identifikovanými vyššie. Preto sme následne pre každú z tried vykonali analýzu súvisiacich implementačných oblastí či funkcií (nazývaných ďalej v práci ako služba), na ktoré má priamy vplyv konkrétne riešenie. Ako inšpirácia slúžili funkcie konkrétnych, v súčasnosti dostupných smerovacích protokolov. Výsledkom našej analýzy sú identifikované nasledovné služby, z ktorých niektoré sú spoločné pre všetky triedy spomínané vyššie, a niektoré sa medzi triedami líšia:

- adresácia susedov

- fragmentácia správ smerovacieho protokolu
- formát správ pozostávajúci z modulárnych štruktúr
- spoľahlivý prenos správ
- priradenie sieťovej adresy k rozhraniu
- identifikátor protokolu (EtherType, Protocol ID)
- zistenie susedom podporovaných sieťových protokolov
- prenos adresy nasledujúceho smerovača (next hop adresy)

Základnou službou nevyhnutnou pre smerovacie protokoly všetkých tried je riešenie ***adresácie***. Použité adresovanie susedov má vplyv na správu susedstiev medzi smerovačmi, ako aj identifikáciu konkrétneho suseda v pracovných databázach smerovacieho protokolu. Jedným spôsobom zabezpečenia identifikácie suseda, nezávislým od identifikovanej triedy integrovanosti, je špeciálne pole v hlavičke smerovacieho protokolu, vyhradené na tento účel. Ako prvok adresácie je možné použiť aj adresu rozhrania (linkovú aj sieťovú), ktorá je ale závislá od typu siete a konkrétnej triedy integrovanosti. V prípade tried 1 a 2 (používajúcich L2 transport) a multiaccess siete ako je Ethernet je možné použiť len linkovú adresu. Ak ide ale o point-to-point sieť, L2 adresy nie sú k dispozícii, čo vedie k použitiu iných identifikátorov, ako napríklad meno rozhrania. V prípade tried 3 a 4 (používajúcich L3 transport) je možné v prípade multiaccess sietí, ako aj point-to-point použiť na identifikáciu suseda sieťovú adresu.

Ďalšou nemenej dôležitou nami identifikovanou službou, ktorú je potrebné riešiť pri prenose, je fragmentácia správ smerovacieho protokolu v prípade MTU menšieho, ako je objem prenášaných smerovacích informácií, potrebných doručiť susedovi. Smerovací protokol musí fragmentáciu riešiť vo vlastnej réžii nezávisle od triedy identifikovanej vyššie.

Ako vyplýva z kapitol 1 a 2, smerovacie protokoly, ktoré boli navrhnuté s dôrazom na nezávislosť od sieťového protokolu a jednoduché rozširovanie, mohli pridať podporu IPv6 bez kompletného prepracovania. Preto je jedna zo zásadných funkcií transportu (v tomto prípade nie je potrebné rozlišovať správy riadenia susedstva a správy prenosu samotnej smerovacej informácie) využitie TLV, alebo iných modulárnych štruktúr, ktoré zaručujú jednoduchosť rozširovania smerovacieho protokolu o nové funkcie v budúcnosti, ako je napríklad podpora nových sieťových protokolov či zavádzanie nových funkcionalít.

Každý smerovací protokol, ktorý vytvára susedské relácie, musí mať zabezpečený spoľahlivý prenos správ smerovacieho protokolu. V prípade vytvárania susedstva pre každý

sieťový protokol osobitne, musí transportný protokol naviac zohľadňovať viaceré relácie a vedieť priradiť prijatý paket ku konkrétnej relácii.

Ďalšou z nevyhnutných služieb je taktiež schopnosť priradiť sieťovú adresu k rozhraniu. V prípade transportu na linkovej vrstve je po aktivácií smerovacieho protokolu na rozhraní potrebné zistiť, ktoré sieťové protokoly sú na konkrétnom rozhraní podporované. Následne je potrebné vytvoriť rôzny počet susedstiev, ktorý sa odvíja od daného protokolu. Podobná situácia nastáva v prípade triedy integrovanosti číslo 4 (L3 transport, jedno susedstvo) kde je taktiež potrebné zistiť podporované adresové rodiny. Následne musí smerovač prislúchajúce smerovacie informácie zaslať susedovi, keďže na rozhraní môžu byť k dispozícii aj iné sieťové protokoly, akým je vytvorené susedstvo. V prípade triedy 3 nie je táto služba potrebná, keďže susedstvo je vytvorené pre každý sieťový protokol, ktorému zodpovedá aj oddelený transport.

Pre transport na linkovej vrstve je potrebné brať do úvahy, v prípade Ethernetu, aj pridelenie EtherType identifikátora, ktorý je pre každý protokol rozdielny. Analogicky pre transport na sieťovej vrstve je potrebný identifikátor Protocol ID.

V prípade transportu na linkovej vrstve, ale aj sieťovej vrstve so spoločným susedstvom je potrebné zistiť od suseda podporované adresové rodiny, aby sme mu neposielali informácie o existencií ciest adresových rodín, ktorých pakety nebude môcť preniesť.

Poslednou z identifikovaných služieb je prenos sieťovej adresy nasledujúceho smerovača (next hop adresy). Niektoré smerovacie protokoly, ako napríklad EIGRP alebo BGP, evidujú adresu nasledujúceho smerovača prislúchajúcu smerovacej informácii na základe adresy suseda, od ktorého túto informáciu prijali. V prípade identifikovanej triedy integrovanosti s číslom 4 (L3 transport, spoločné susedstvo) nastáva problém, keďže v jednom susedstve prenášame smerovacie informácie viacerých sieťových protokolov. Preto je potrebné, aby sme spolu so smerovacími záznamami preniesli aj next hop adresy, keďže nemusia zodpovedať susedstvu. Podobný problém nastáva aj v prípade tried integrovanosti s číslom 1 a 2 (L2 transport), kde vôbec nie je zo susedstva možné zistiť sieťovú adresu. Preto je ich nutné preniesť spolu so smerovacími informáciami, prípadne definovať osobitnú TLV štruktúru vhodnú na prenos tejto informácie.

3 Formálne popisy smerovacích protokolov

Pri návrhu smerovacích, ale aj iných sieťových protokolov zohráva dôležitú úlohu korektný návrh komplexného systému, akým protokoly bezpochyby sú, a následná verifikácia vytvoreného návrhu, ktorá pomáha odstrániť prípadné zraniteľnosti, nepresnosti a zlepšuje efektivitu navrhovaného protokolu. Pre čitateľa je zaiste zrejmé, že chyby navrhovaného protokolu je ťažšie odstrániť počas implementačnej fázy a neskôr počas testovania výslednej implementácie, ako v počiatočných fázach zameraných na dizajn protokolu. Preto rôzne zdroje odporúčajú už počas návrhovej a verifikačnej časti vývoja protokolu využiť niektoré z nástrojov a metód uľahčujúcich kontrolu dizajnu navrhovaného protokolu.

Jedným zo spôsobov, ktorý napomáha pri korektnom návrhu systémov je použitie formálnych metód (angl. *Formal Description Techniques - FDT*). Základom týchto metód je notácia, ktorá reprezentuje jednoznačnú špecifikáciu navrhovaného protokolu a jeho podporných nástrojov, a za pomoci matematiky umožňuje skúmať vlastnosti návrhu. Tam, kde prirodzený jazyk (angličtina, slovenčina...) môže spôsobiť nejednoznačnosť, formálne metódy ponúkajú prostriedky na predchádzanie takýchto nechcených javov. Samotné použitie formálnych metód nezaručuje korektnosť návrhu, avšak ich správna aplikácia zvyšuje porozumenie požiadavkám na návrh systému [10]–[12].

Medzi najznámejšie formálne metódy používané na modelovanie a verifikáciu protokolov patria: Konečné stavové automaty (angl. *Finite State Machines - FSM*) a Petriho siete [10].

3.1 Konečné stavové automaty

Konečné stavové automaty pozostávajú zo súboru stavov a prechodových pravidiel medzi stavmi. Môžu byť reprezentované grafom prechodov medzi stavmi, príkladom je graf algoritmu DUAL použitého v EIGRP (Obrázok 5). Zvyčajne sú ale konečné stavové automaty špecifikované vo forme tabuľky. Vzorová prechodová tabuľka pre konečný stavový automat typu Mealy je znázornená dole (Tabuľka 1) [10], [13].

Každému stavu konečného stavového automatu môže prislúchať viacero prechodových pravidiel, pričom jeden riadok tabuľky špecifikuje práve jedno prechodové pravidlo. Vzorová tabuľka obsahuje množinu stavov {q0, q1, q2, q3}. Prechodové pravidlo

pozostáva zo štyroch častí, každá časť je jeden stĺpec tabuľky. Prvé dve časti obsahujú podmienky, ktoré musia byť v prípade zmeny stavu splnené. Druhé dve časti definujú následok zmeny stavu. V tradičných konečných stavových modeloch je prostredie reprezentované konečnými množinami vstupných a výstupných hodnôt [13].

Podmienka zmeny stavu		**Následok zmeny stavu**	
Aktuálny stav	**Vstup**	**Výstup**	**Nasledujúci stav**
q0	-	1	q2
q1	-	0	q0
q2	0	0	q3
q2	1	0	q1
q3	0	0	q0
q3	1	0	q1

Tabuľka 1: Konečný stavový automat

3.2 Petriho siete

Petriho siete prvýkrát spomenul vo svojej dizertačnej práci nemecký profesor Carl Adam Petri [14] v roku 1962, ktorou ukončil štúdium na univerzite v Bonne. Sú grafickým a matematickým nástrojom na modelovanie, analýzu a verifikáciu distribuovaných systémov a súbežných procesov v rôznych oblastiach, ako napríklad umelá inteligencia, rôzne paralelné systémy, riadiace a komunikačné systémy, výrobné systémy, číslicová analýza, počítačové siete a iné. Vo všeobecnosti sú Petriho siete vhodný nástroj na opis a analýzu paralelných, asynchrónnych, distribuovaných, nedeterministických, alebo aj stochastických systémov. Modelovanie a analýza systému sa zvyčajne vykonáva pomocou teoretického základu Petriho sietí, zatiaľ čo vizualizácia už namodelovaného systému prebehne pomocou grafickej reprezentácie. Táto kombinácia matematickej teórie a grafickej vizualizácie je hlavným dôvodom úspechu Petriho sietí [15]–[17].

Petriho sieť patrí medzi modelovo-orientované prístupy formálnych metód, ktoré modelujú stav systému a jeho zmeny. Formálne je definovaná ako bipartitný orientovaný

graf pozostávajúci z dvoch typov uzlov, ktoré sa nazývajú miesta a prechody, a sú navzájom prepojené hranami [18]–[20].

Formálny zápis Petriho siete sa v literatúre uvádza aj ako:

$$N = (P, T, A)$$

kde P a T sú disjunktné konečné množiny miest a prechodov a

$$A \subseteq (P * T) \cup (T * P)$$

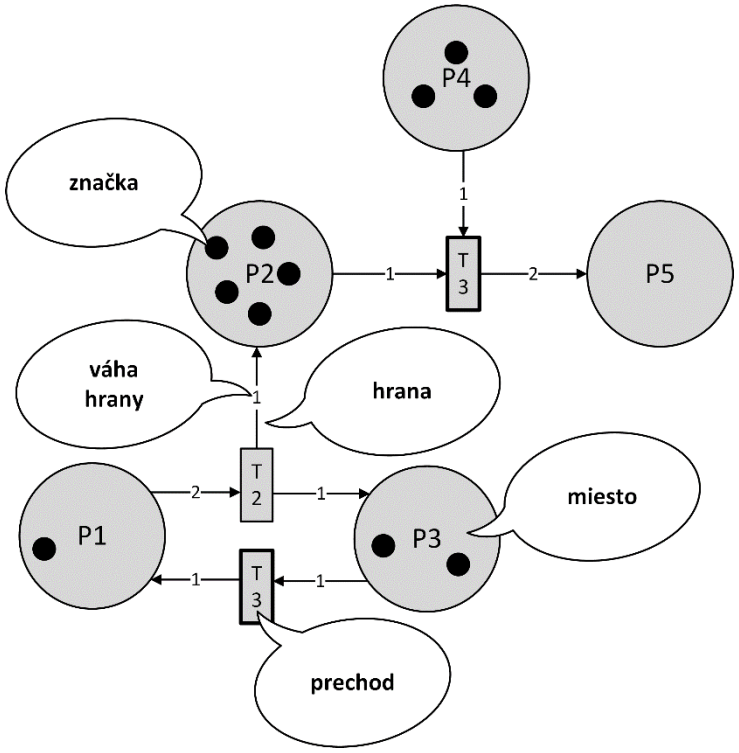
kde $(\forall t \in T)\ (\exists p, q \in P)\ (p, t), (t, q) \in A$ je množina hrán. Každý prechod môže byť hranami spojený s nezáporným počtom vstupných a výstupných miest. Každé miesto môže obsahovať nezáporný počet značiek reprezentujúcich kapacitu miesta a každá hrana má priradené nezáporné číslo reprezentujúce jej váhu. Prechod sa môže vykonať len v prípade, ak je aktívny, čo znamená, že sú splnené všetky nasledujúce podmienky:

- Počet značiek na vstupnom mieste je väčší alebo rovný váhe hrany vstupujúcej do prechodu
- Kapacita výstupného miesta je väčšia alebo rovná váhe hrany vychádzajúcej z prechodu.

V prípade štúdia zmien stavu modelovaného systému je každému miestu priradené nezáporné celé číslo, ktoré vyjadruje počet značiek v tomto mieste. Ak systém študujeme z pohľadu podmienok a udalostí, miesta reprezentujú podmienky a prechody reprezentujú udalosti. Značka prítomná v určitom mieste je interpretovaná ako platnosť podmienky. Príklad grafickej reprezentácie jednoduchej Petriho siete je znázornený na obrázku (Obrázok 2) [11], [15], [17].

Vzorová Petriho sieť, znázornená na obrázku (Obrázok 2), pozostáva z troch miest, ktoré sú reprezentované kruhmi so šedým pozadím, s označeniami *P1*, *P2* a *P3*, troch štvorcov so sivým pozadím a označením *T1*, *T2* a *T3*, ktoré reprezentujú prechody a nakoniec šípkami prepájajúcimi prechody a miesta reprezentujúcimi hrany s váhami v strede čiary. Niektoré miesta obsahujú nezáporný počet značiek, reprezentovaných čiernymi kruhmi. Prechody, ktoré sú aktívne, majú orámovanie zvýraznené hrubou čiarou. Ako príklad môžeme uviesť prechod *T3*, ktorý je aktívny preto, že vstupné miesto obsahuje dve značky a vstupná hrana má váhu taktiež dva. Výstupné miesto nemá uvedenú maximálnu kapacitu a výstupná hrana má váhu jeden. Za povšimnutie stojí aj fakt, že

prechod *T2* nie je aktivovaný. To je spôsobené iba jednou značkou na vstupnom mieste a váhou vstupnej hrany s hodnotou dva. Avšak po vykonaní prechodu *T3* je pridaná značka do miesta *P1*, čo spôsobí splnení vstupných podmienok pre prechod *T2*, ktorý bude následne aktívny.



Obrázok 2: Jednoduchý graf Petriho siete, zdroj autor

Keďže Petriho siete sa používajú už dlhú dobu v rozličných odvetviach, rozlišujeme množstvo rozšírení a modifikácií základnej koncepcie Petriho sietí. Niektoré z modifikácií sú spätne kompatibilné, čo znamená, že môžu byť prepísané pomocou notácie pôvodných Petriho sietí. Ako príklad môžeme uviesť farbené Petriho siete. Avšak niektoré rozšírenia pridávajú prvky a vlastnosti, ktoré nie je možné prepísať pomocou pôvodnej notácie bez straty prvkov modelu. Príkladom nekompatibilného rozšírenia sú časové Petriho siete [19], [21], [22].

Niektoré z najpopulárnejších vlastností, ktoré nie sú k dispozícii v pôvodných Petriho sieťach sú:

- značka s priradenou hodnotou – značky v pôvodných Petriho sieťach sú nerozlíšiteľné
- hierarchia v modeli – napríklad značky môžu obsahovať ďalšie Petriho siete a tak vytvárať vnorené modely

- priradenie času prechodom – čas každého prechodu môže byť deterministický alebo nedeterministický
- rôzne typy hrán.

Viaceré dostupné zdroje [10], [12], [15], [23]–[26] uvádzajú ako vhodný nástroj na modelovanie sieťových protokolov, vrátane smerovacích protokolov, z prostredia formálnych metód, farbené Petriho siete (angl. *Colored Petri nets - CPN*), ktoré zaraďujeme medzi Petriho siete vyššej úrovne (angl. *High-level Petri nets*) [11].

3.3 Farbené Petriho siete

Farbené Petriho siete (CPN) je modelovací jazyk, ktorý kombinuje užitočné funkcie vysokoúrovňových programovacích jazykov a matematického základu poskytovaného Petriho sieťami. Grafická notácia a prvky potrebné pre modelovanie súbežných procesov, synchronizáciu a komunikáciu sú zabezpečované Petriho sieťami. Prvky vysokoúrovňových programovacích jazykov poskytuje CPN ML čo je funkcionálny jazyk odvodený od všeobecne používaného programovacieho jazyka Standard ML (SML). Medzi prvky poskytované CPN ML jazykom môžeme zaradiť dátové typy, nástroje na vytváranie základných, ako aj parametrických modelov a taktiež prvky nevyhnutné pre popis manipulácie s údajmi. Vďaka funkciám popísaným vyššie, sa farbené Petriho siete zaraďujú medzi siete vyššej úrovne [12], [24], [26]–[30].

Formálne sú farbené Petriho siete definované ako:

$$N = (P, T, A, C, V, CF, GF, AF, IF)$$

kde:

- P a T sú konečné disjunktné množiny miest a prechodov.
- $A \subseteq (P * T) \cup (T * P)$ je množina orientovaných hrán.
- C je konečná množina farieb.
- V je množina premenných s určeným typom, pričom typ každej premennej je z množiny C.
- $CF: P \rightarrow C$ je funkcia, ktorá každému miestu z množiny P priradí farbu z množiny C.
- $GF: T \rightarrow Expression$ je funkcia, ktorá každému prechodu z množiny T priradí výraz obmedzujúci prípady, kedy je prechod aktívny.
- $AF: A \rightarrow Expression$ je funkcia, ktorá priradí výraz každej hrane z množiny A.
  $IF: P \rightarrow Expression$ je funkcia, ktorá každému miestu z množiny P priradí inicializačný výraz.

Veľkou výhodou farbených Petriho sietí je, že ich je možné použiť na modelovanie rôznych typov systémov, ktoré pozostávajú zo súbežných procesov. Vďaka tomu, že farbené Petriho siete neboli vyvinuté pre riešenie len úzkej skupinky problémov sú zaraďované medzi univerzálne modelovacie jazyky. Medzi najčastejšie uvádzané použitie farbených Petriho sietí patrí modelovanie distribuovaných algoritmov, vstavaných systémov, dátových sietí a v neposlednom rade aj komunikačných protokolov. Avšak je ich možné použiť aj na modelovanie všeobecnejších a aj netechnických systémov, ako sú napríklad systémy výroby, obchodné ako aj manažérske procesy, systémy agentov alebo aj modely obchodných činností [12], [15], [23], [24], [26], [27], [30].

Výhodou farbených Petriho sietí, na rozdiel od klasických Petriho sietí, je podpora hierarchie, farby a času v modeli. Hierarchia umožňuje skladanie modelov z viacerých menších modulov, podobne ako hierarchia použitá v objektovo-orientovaných programovacích jazykoch. Výhodou hierarchického princípu je aj znovupoužiteľnosť modulov, prípadne opakovanie modulov v rámci modelovaného systému. Podpora času v modeli nám umožňuje vyznačiť, ako dlho trvá určitý prechod v systéme. Ďalšou z výhod farbených Petriho sietí je použitie malého množstva prvkov, z ktorých je možné vytvoriť komplexné siete, a taktiež veľké množstvo softvérových nástrojov na kreslenie, simuláciu a formálnu analýzu celej siete. [10], [15], [23].

Verifikáciu už vytvoreného modelu je možné opísať ako proces, ktorý dôkladne skontroluje implementáciu modelu a taktiež zisťuje, či bol namodelovaný podľa špecifikácie. Pomocou verifikácie je teda možné dokázať, že určité požadované vlastnosti sú súčasťou vytvoreného modelu a naopak nežiadúce prvky modelu nie sú prítomné. Na verifikáciu vytvoreného modelu využívajú farbené Petriho siete analýzu stavového priestoru (ang. State space analysis) [12], [23], [24], [26].

Mechanizmus analýzy stavového priestoru vznikol na základe veľmi jednoduchej myšlienky. Najskôr je potrebné vypočítať všetky dosiahnuteľné stavy, v ktorých sa môže modelovaný systém nachádzať spolu so zmenami stavov. Po dokončení výpočtov je vytvorený orientovaný graf zo všetkých dosiahnuteľných stavov, pričom jeho uzly reprezentujú stavy a orientované hrany reprezentujú udalosti. Veľkou výhodou farbených Petriho sietí je, že stavový priestor je vytvorený úplne automaticky a bez potreby zásahu používateľa. Vytvorený graf stavového priestoru nám umožňuje zodpovedať otázky a objasniť nejasnosti ohľadom správania namodelovaného systému. Ako príklad je možné uviesť dosiahnuteľnosť zvoleného stavu modelu, prítomnosť miest, v ktorých nastáva

uviaznutie systému alebo garantovanie doručenia správ modelovaným protokolom [12], [23], [26], [27], [30].

Postupom času vznikli viaceré nástroje na vytvorenie modelu farbených Petriho sietí pomocou grafickej reprezentácie (systémom drag-and-drop). Tieto nástroje zvyčajne umožňujú používateľovi vykonať simuláciu vytvoreného modelu, ako aj verifikáciu a formálnu analýzu celého systému. Aj napriek tomu, že základom farbených Petriho sietí sú formálne matematické princípy, literatúra [12], [24], [26]–[28] uvádza, že pre praktické použitie softvérových nástrojov na podporu modelovania farbených Petriho sietí postačuje pochopenie a sémantika farbených Petriho sietí a taktiež dobrá znalosť syntaxe modelovacieho jazyka. Za zmienku stojí aj fakt, že táto vlastnosť farbených Petriho sietí je podobná s vysokoúrovňovými programovacími jazykmi, ako sú Java, Python alebo C. Tieto jazyky sú veľmi rozšírené a úspešne používané programátormi aj napriek tomu, že nie vždy poznajú formálnu definíciu jazyka, ktorý dennodenne používajú. Preto je možné sa naučiť a úspešne vytvárať modely farbených Petriho sietí pomocou dostupných softvérových nástrojov aj bez štúdia všetkých formálnych princípov, ktoré poskytujú základ farbených Petriho sietí [12], [23], [24], [26], [28], [30].

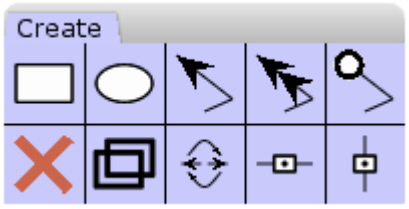
Za vytvorením farbených Petriho sietí stojí skupina výskumníkov z univerzity Aarhus v Dánsku, ktorá je považovaná za priekopníkov v teoretických základoch ako aj aplikácii vysokoúrovňových Petriho sietí. Ich výsledkom je taktiež publikácia veľkého množstva vedeckých článkov a kníh, v ktorých sa pojednáva hlavne o formálnych základoch farbených Petriho sietí ako aj ich aplikácii v praxi a vo výučbe. Hlavnými členmi skupiny boli profesori Kurt Jansen, Lars. M. Kristensen a Soren Christensen [31].

3.4 CPN Tools

CPN Tools (Colored Petri Nets Tools) je softvérový nástroj pôvodne vyvinutý výskumníkmi z univerzity Aarhus v Dánsku počas rokov 2000 až 2010. Na softvéri pracovala skupina CPN pod vedením profesorov Kurt Jansen a Lars M. Kristensen. Avšak v roku 2010 sa výskumná skupina rozpadla, keďže sa vedúci tímu začali zaoberať inými témami. Preto správu programu prevzal tím z technickej univerzity Eindhoven v Holandsku [12], [23], [31], [32].

Hlavným účelom programu je poskytnúť používateľovi všetky nástroje nevyhnutné na vytváranie modelov farbených Petriho sietí pomocou priamej manipulácie s grafickým

pohľadom na model pomocou drag-and-drop systému. Všetky nástroje v grafickom rozhraní programu sú organizované do takzvaných paliet. Ako príklad môžeme uviesť paletu s názvom Create, ktorá obsahuje základné prvky na vytvorenie štruktúry modelu farbenej Petriho siete ako sú miesta, prechody, hrany, ale aj pomocné nástroje na zmazanie a klonovanie existujúcich prvok modelu alebo vytváranie pomocných čiar. Paleta Create je znázornená na obrázku (Obrázok 3).



Obrázok 3: Paleta nástrojov s názvom Create, zdroj autor

Ďalšia sada nástrojov, organizovaná v palete s názvom hierarchia, umožňuje používateľovi vytvárať modely pozostávajúce z viacerých spolu pospájaných menších modulov. Tieto nástroje taktiež umožňujú vytvárať menšie časti, ktoré môžu byť opäť použiteľné ako súčasť iných modelov. Z popisu je zrejmé, že tento princíp je podobný triedam, inštanciám a rozhraniam použitým v objektovo-orientovanom programovaní. Návrhár výsledného modelu môže taktiež vďaka týmto nástrojom používať rôzne prístupy návrhu modelovaných systémov ako sú top-down alebo bottom-up [24], [26], [32], [33].

Jednou z veľmi užitočných funkcií, na ktoré sú zvyknutí používatelia integrovaných vývojových prostredí (ang. Integrated Development Environment - IDE) pre vysokoúrovňové programovacie jazyky je syntaktická kontrola písaného zdrojového kódu. CPN Tools rovnako vykonávajú syntaktickú a taktiež typovú kontrolu inkrementálne a aj počas úpravy zdrojového kódu. To znamená, že je možné vykonať aj časti modelu (keďže nad nimi už prebehla syntaktická kontrola) bez toho, aby bol dokončený celý model [10], [23], [24], [26], [27], [32].

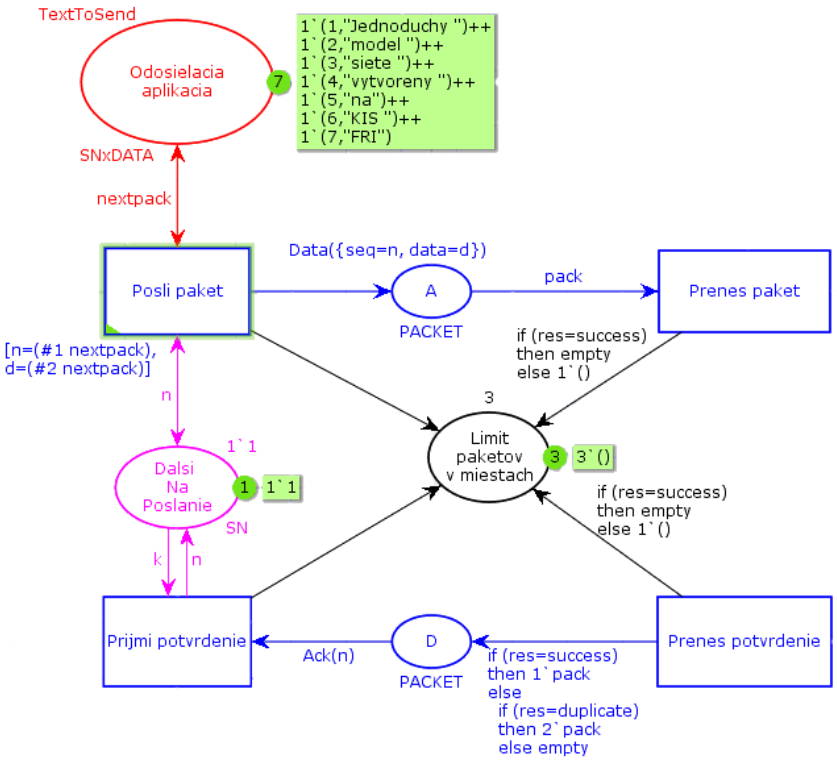
Ďalšia z funkcií vykonávaných inkrementálne, počas vytvárania a úpravy modelu je generovanie kódu pre spustenie simulácie. Vygenerovaný kód je následne použitý simulačným jadrom programu CPN Tools, ktorý používateľovi umožní získať užitočné informácie o vytvorenom modeli. Simulačné jadro je schopné spúšťať simulácie aj veľkých hierarchických modelov farbených Petriho sietí, pretože využíva pokročilé dátové štruktúry a efektívne algoritmy. Počet simulačných krokov taktiež nezávisí od počtu miest a prechodov v modeli, čo značne zefektívňuje simuláciu. Simulácia v nástroji CPN Tools je

veľmi flexibilná, keďže používateľovi umožňuje interagovať s bežiacou simuláciou buď zastavením a prezeraním stavu modelu po každom kroku, alebo po určitom zadanom počte krokov. Udalosti, ako napríklad zmeny aktívnych prechodov počas simulácie sú zobrazené priamo na grafickej reprezentácii modelu. V interaktívnom režime simulácie (špeciálne vykonanie jedného kroku simulácie) je taktiež možné vybrať, ktorý prechod sa v aktuálnom kroku vykoná a tak potencionálne preskúmať hraničné stavy namodelovaného systému. Avšak v automatickom režime simulácie (keď je simulácia zastavená až po určitom počte krokov) je priradenie hodnôt premenným a výber prechodov vykonaný náhodne. Všetky tieto nástroje sú organizované v palete simulácie [10], [12], [15], [23], [24], [26], [32].

Keďže simulácia je vhodná na detekciu chýb v modeli, ale nezaručuje správnosť modelu ako celku, a teda aj výsledného protokolu, CPN Tools obsahujú sadu nástrojov slúžiacu na verifikáciu skonštruovaného modelu za účelom potvrdenia správnosti návrhu. Verifikácia je vykonávaná pomocou vypočítania celého stavového priestoru. Používateľ sa môže rozhodnúť, či chce zobraziť kompletnú grafickú reprezentáciu stavového priestoru (vhodné len pri malých modeloch), alebo prehliadať len časti stavového priestoru aby prekontroloval význačné a kľúčové časti navrhnutého systému. Avšak, keďže celý stavový priestor je počas výpočtu uložený v operačnej pamäti, kompletný stavový priestor môže byť vypočítaný len v prípade použitia stroja s dostatočnou kapacitou RAM pamäte. Ďalšou z nevýhod verifikačných nástrojov programu CPN Tools je využívanie len jedného jadra procesora, čo limituje výpočtový výkon veľkých stavových priestorov. Nástroje na generovanie a prehliadanie stavového priestoru sú dostupné v palete s názvom State Space [10], [12], [23]–[26], [32].

Softvérový nástroj CPN Tools je dostupný len pre operačný systém Windows, pričom je možné bezplatne ho stiahnuť na oficiálnej stránke [32].

Na obrázku (Obrázok 4) je znázornený príklad časti grafickej reprezentácie jednoduchej farbenej Petriho siete, namodelovanej pomocou nástroja CPN Tools.



Obrázok 4: Príklad modelu jednoduchej farbenej Petriho siete, zdroj autor

Ako môžeme vidieť na obrázku (Obrázok 4), grafická reprezentácia farbenej Petriho siete prebrala mnohé dizajnové prvky z reprezentácie originálnych Petriho sietí. Miesta sú znázornené ako elipsy a prechody majú tvar obdĺžnika. Hrany sú reprezentované pomocou orientovaných šípok. Za zmienku stojí aj fakt, že v modeli znázornenom na obrázku sa nevyskytujú hrany prepájajúce priamo dve miesta alebo priamo dva prechody medzi sebou, pretože by bol porušený základný princíp originálnych Petriho sietí, uvedený v časti 3.2 (hrana môže prepájať len miesto s prechodom). Na obrázku je aj vidieť, že dizajnér môže v nástroji CPN Tools rozlíšiť jednotlivé prvky modelu rôznou farbou (nie je to ale farba, teda typ premennej vo farbených Petriho sieťach) pre sprehľadnenie modelu. Napríklad súvisiace prvky modelu, alebo skupiny prvkov môžu byť zvýraznené rovnakou farbou [10], [12], [15], [25], [29], [32].

Na rozdiel od štandardných Petriho sietí majú modely farbených Petriho sietí dva a viac nápisov v blízkosti miesta. Prvý, ktorý je zvyčajne umiestnený nad miestom reprezentuje výraz alebo premennú, ktorú nadobúda značka, ktorá môže byť v mieste. Nápis, zvyčajne umiestnený pod miesto reprezentuje farbu (a teda dátový typ) značky, ktorá sa môže nachádzať v mieste. Nápis, ktorý je vo vnútri elipsy reprezentujúcej miesto, označuje jeho názov a mal by byť samo popisný, teda z názvu by malo byť čitateľovi modelu jasné,

akú funkciu miesto zastáva. Prechody sú označované podobným spôsobom ako miesta. Zelená značka, ktorá je zvyčajne umiestnená na pravej strane miesta predstavuje hodnotu alebo viaceré hodnoty značiek, ktoré sa nachádzajú v mieste v aktuálnom kroku simulácie vytvoreného modelu. Nápis v blízkosti hrany predstavuje podmienku, ktorej splnenie aktivuje prechod za predpokladu, že aj podmienky na všetkých ostatných hranách vedúcich od a z prechodu sú splnené. Aktivovaný prechod je znázornený zeleným trojuholníkom v ľavom dolnom rohu a je taktiež podfarbený zelenou žiarou okolo celého obdĺžnika reprezentujúceho prechod [12], [15], [16], [24], [26], [29], [30], [32], [34].

3.5 Prístupy modelovania sieťových protokolov

Literatúra uvádza viaceré konceptuálne prístupy vytvárania sieťových protokolov ale aj iných súbežných a zložitých systémov. Táto kapitola rozoberá dva najpoužívanejšie a najznámejšie prístupy: top-down a bottom-up prístup.

Top-down prístup, ktorý je často označovaný aj ako dekompozícia, začína s namodelovaním systému ako celku, ktorý je ale veľmi všeobecný a vôbec nezasahuje do detailov návrhu. Potom sú postupne do modelu zapracovávané dôležité prvky s požadovanou úrovňou detailov. To znamená, že postupom času sú niektoré významné časti namodelované do úplných detailov a menej podstatné časti zostávajú úplne alebo čiastočne bez detailov. Týmto spôsobom získava dizajnér náhľad do kompozičných pod častí modelu [12], [33], [35], [36].

Prístup bottom-up postupuje úplne opačným spôsobom ako top-down. Začína s modelovaním určitých pod modulov systému do úplných detailov. Akonáhle je dizajnér spokojný s úrovňou detailov každého pod modulu, začína ich prepájať a vytvárať z nich rozsiahly model. Preto sa tento prístup často prirovnáva k spôsobu rastu rastlín zo semienok, keďže postupne sa z malého semienka po čase stane veľká rastlina. Avšak, tento prístup so sebou prináša riziko v podobe prílišného sústredenia sa na detaily, ktoré v niektorých prípadoch nemusia byť nevyhnutné a zabúdanie na systém ako celok. To môže viesť k prílišnej optimalizácii pod modulov namiesto smerovania k systému ako celku [12], [26], [37].

Viaceré zdroje [12], [18], [26], [28], [33], [35]–[38] ale uvádzajú, že dizajnér modelu komplexného systému by mal poznať výhody a nevýhody oboch prístupov a vybrať si prístup, ktorý sa najviac hodí na konkrétny modelovaný systém na základe jeho

predchádzajúcich skúseností s modelovaním iných systémov. Avšak vo väčšine prípadov dizajnéri kombinujú do určitej miery oba prístupy aby využili výhody každého s prístupov a vyhli sa ich slabým stránkam. Napríklad je možné začať s málo detailným modelom popisujúcim celý systém, potom pridať niektoré vhodné pod moduly, ktoré boli už namodelované do detailov v inom systéme. Potom môže dizajnér pridávať detaily do určitých častí modelu a tak postupne zlepšovať detailnosť celého systému. Môže sa ale stať, že model ako celok neobsahuje požadovanú funkcionalitu, a preto dizajnér pridá potrebnú časť, ktorú namodeluje opäť kombináciou oboch prístupov.

4 Enhanced Interior Gateway Routing Protocol

Ako bolo spomenuté v úvode, smerovací protokol Enhanced Interior Gateway Routing Protocol (EIGRP) je pre túto prácu dôležitý pre svoje unikátne vlastnosti, ako aj fakt, že bol vybratý ako vzor pre vytvorenie modelu tretej triedy riešenia protokolovej integrovanosti. V tejto kapitole sa preto bližšie pozrieme na vlastnosti tohto protokolu.

EIGRP je, podobne ako RIP, interný smerovací protokol typu distance-vector, ale obsahuje niektoré pokročilé funkcie. Bol navrhnutý spoločnosťou Cisco Systems a vznikol úplným prepracovaním proprietárneho protokolu IGRP, ktorý bol uvedený na trh v roku 1985, taktiež spoločnosťou Cisco Systems. EIGRP s ním tiež zdieľa maximálny počet hopov, použitie kompozitnej metriky a podporu rôznych sieťových protokolov, vrátane IPv4 aj IPv6. EIGRP podobne ako OSPFv3 v prípade prevádzky oboch sieťových protokolov vytvára susedstvá pre každý protokol osobitne, ale ako transport je pre IPv4 smerovacie informácie použitý IPv4 protokol a pre IPv6 smerovacie informácie sú vytvárané IPv6 pakety. EIGRP bol pôvodne, rovnako ako IGRP, proprietárny protokol, preto bol dostupný len na zariadeniach vyrábaných Cisco-m. Avšak v roku 2013 padlo rozhodnutie uvoľniť základnú špecifikáciu EIGRP, ktorá bola neskôr v roku 2016 publikovaná ako RFC 7868, čo umožnilo vytvorenie rôznych open-source implementácií tohto protokolu [4], [39].

V porovnaní s inými distance-vector smerovacími protokolmi, EIGRP je navrhnutý na prevádzku v rozsiahlych sieťach, podporuje výmenu smerovacích informácií s variabilnou sieťovou maskou, autentifikáciu posielaných správ a vyniká veľmi rýchlou konvergenciou. V protiklade s RIP posiela susedom rozdielové aktualizácie a nie periodicky celé smerovacie tabuľky, čo je možné vďaka vytváraniu susedstiev s okolitými smerovačmi a použitiu vlastného transportného protokolu Reliable Transport Protocol (RTP), ktorý podporuje spoľahlivé unicastové aj multicastové doručovanie správ. Preto je EIGRP v niektorej literatúre nesprávne označovaný ako hybridný smerovací protokol, keďže toto správanie sa výrazne podobá na smerovacie protokoly typu link-state [40].

Unikátnou vlastnosťou, ktorú neponúka žiadny iný smerovací protokol, je garancia bezslučkovej činnosti za predpokladu správnej konfigurácie. Na zistenie, či trasa do cieľa zaručene neobsahuje slučku, využíva EIGRP podmienku s názvom Feasibility Condition (FC). Ďalšou zaujímavou vlastnosťou EIGRP sú difúzne výpočty, ktoré sú riadené algoritmom Diffusing Update Algorithm (DUAL), ktorý bol vytvorený v roku 1989 výskumníkom J. J. Garcia-Luna-Aceves. V prípade, že sa v súčasnosti najlepšia cesta do

cieľovej siete stane neplatnou pre porušenie FC, smerovač sa musí opýtať susedov na aktualizovanú najlepšiu cestu do cieľovej siete. Smerovače, ktoré nie sú ovplyvnené touto zmenou odpovedajú okamžite, ale tie, ktoré sú ovplyvnené, buď okamžite zmenia prislúchajúce položky v smerovacej tabuľke, alebo sa pýtajú svojich susedov. Týmto spôsobom sa otázky rozšíria po zasiahnutej časti siete, čo je princípom difúznych výpočtov. DUAL je zodpovedný nielen za riadenie difúznych výpočtov, ale aj spracovanie prijatých smerovacích informácií, a taktiež prispieva k rýchlej konvergencii [3], [41], [42].

4.1 Vlastnosti popísané v RFC 7868

Súčasťou uvoľnenia špecifikácie EIGRP v roku 2013 bolo aj zverejnenie IETF návrhu, ktorý bol neskôr transformovaný do RFC 7868 [41]. Avšak nie všetky vlastnosti a funkcionalita pôvodnej implementácie EIGRP od spoločnosti Cisco boli zverejnené. V nasledujúcich podkapitolách sa budeme zaoberať zverejnenou špecifikáciou [41].

4.1.1 Diffusing Update Algorithm

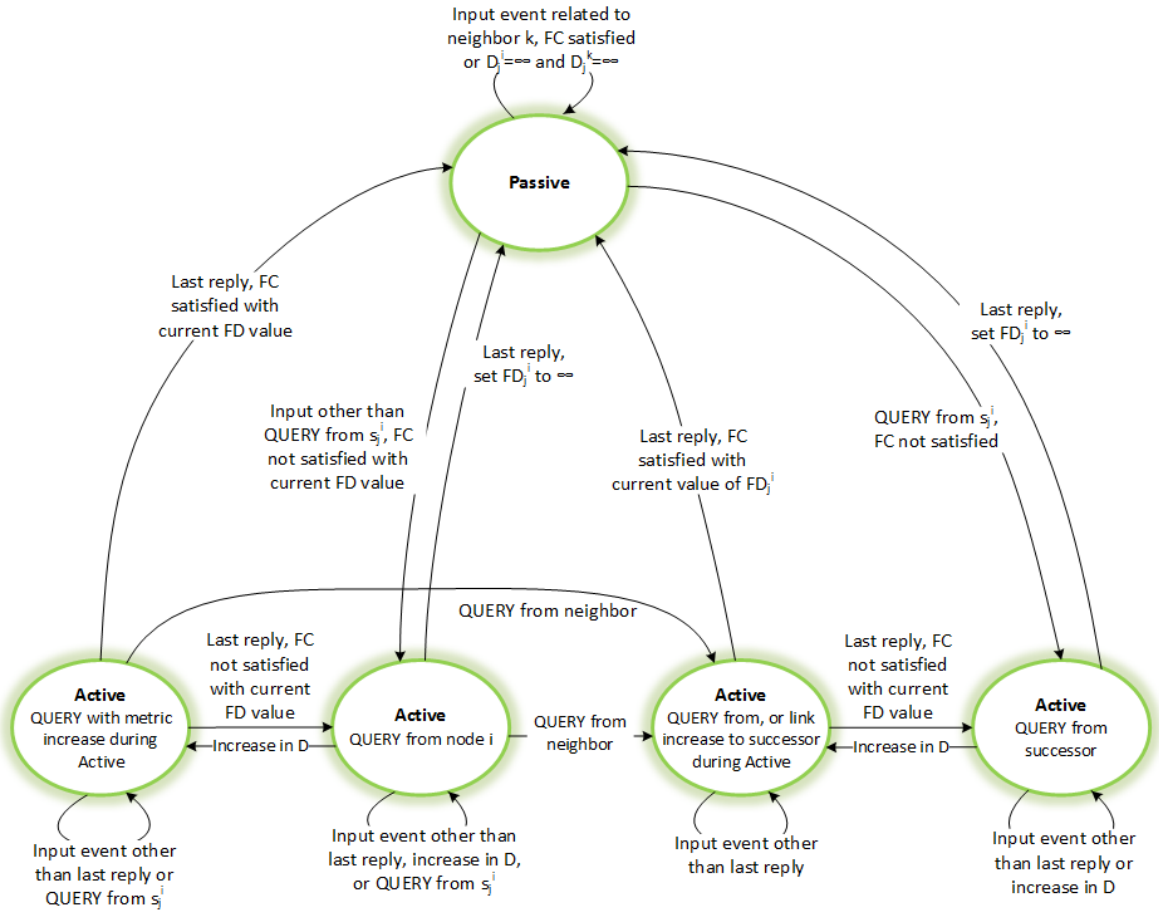
Ako už bolo spomenuté v kapitole 2, DUAL slúži na výpočet ciest s najmenšou metrikou do cieľových sietí, pričom garantuje bezslučkovosť týchto ciest aj počas zmien v sieťovej topológii. Má na starosti aj riadenie difúznych výpočtov, ktoré prispievajú k zníženiu času konvergencie, a taktiež réžiu smerovacieho protokolu.

Smerovací záznam sa môže z pohľadu algoritmu DUAL nachádzať v dvoch stavoch: aktívny a pasívny. V aktívnom stave sa smerovač priamo podieľa na difúznom výpočte, pretože v dôsledku zmeny topológie sused s najnižšou metrikou do cieľovej siete už nespĺňa FC. Ak je sieť v aktívnom stave aj naďalej, je garantované, že neobsahuje slučku, ale nemusí byť najkratšia, alebo vôbec použiteľná. Cesta sa nachádza v pasívnom stave, ak je k dispozícii aspoň jeden sused, trasa cez ktorého spĺňa FC [41], [42].

Dokument [42] uvádza viaceré verzie FC, avšak verzia použitá v EIGRP protokole využíva takzvanú Feasible Distance (FD), ktorá je definovaná pre každú cieľovú sieť ako najmenšia známa vzdialenosť do cieľa od posledného prechodu cesty z aktívneho do pasívneho stavu. FD teda nie je aktuálna vzdialenosť, ale skôr najnižšia zaznamenaná vzdialenosť od zmeny stavu cesty. Aktuálnu vzdialenosť priamo pripojeného susedného smerovača do konkrétnej cieľovej siete reprezentuje hodnota s názvom Reported Distance (RD). Feasibility Condition sa pomocou týchto konceptov definuje nasledovne: „Cesta do konkrétnej siete cez suseda, kde je splnená podmienka RD < FD zaručene neobsahuje

slučku.“ Sused, ktorý spĺňa FC je označený ako Feasible Successor, a ak má naviac cesta cez tohto suseda najnižšiu vzdialenosť aj so započítaním linky medzi smerovačom a susedom, je susedný smerovač označený ako Successor. Každá cesta, ktorá spĺňa FC, je zaručene bez slučky, ale nie všetky cesty bez slučky spĺňajú FC, preto sa FC označuje aj ako dostatočná ale nie nevyhnutná podmienka bezslučkovosti [41]–[43].

Všetky rozhodovacie procesy v EIGRP sú riadené algoritmom DUAL, ktorý je reprezentovaný konečným stavovým automatom (angl. *Finite State Machine - FSM*). Spracováva všetky smerovacie informácie od susedov, a metriku obsiahnutú v aktualizáciách využíva na vybratie ciest, ktoré neobsahujú slučku. Algoritmus spracováva každú cieľovú sieť v topologickej tabuľke osobitne, ale niektoré zmeny v topológii môžu ovplyvniť viacero ciest, ktorých stav sa následne zmení na aktívny. Kompletný diagram konečného automatu DUAL je znázornený na obrázku (Obrázok 5) [41], [43].



Obrázok 5: Konečný stavový automat DUAL, zdroj autor

4.1.2 EIGRP pakety

Aj napriek tomu, že EIGRP podporuje viaceré sieťové protokoly, zverejnené RFC popisuje len IPv4 a IPv6. EIGRP správy sú v špecifikácii ako aj iných zdrojoch označované

ako pakety aj napriek faktu, že sa nejedná o pakety sieťového protokolu. EIGRP pakety sa vkladajú priamo do IP paketu, fragmentácia podporovaná nie je, preto sa ich dĺžka odvíja od MTU na konkrétnom rozhraní, cez ktoré paket odchádza. Ak je potrebné preniesť viac informácií než je veľkosť MTU, je poslaných viacero paketov tak, aby jeden MTU nepresahoval. Protokolové číslo v IP hlavičke je nastavené na 88 pre EIGRP prenášané v IPv4 aj IPv6. Zdrojová adresa v pakete je vždy adresa rozhrania, cez ktoré EIGRP paket odchádza. Cieľová adresa je buď unicast, alebo jedna z nasledujúcich multicast adries: 224.0.0.10 alebo FF02::A.

EIGRP rozlišuje sedem typov paketov, pričom medzi základné patria:

Hello pakety sú využívané na objavenie suseda a udržanie už vytvorenej susedskej relácie. Štandardne sú posielané pomocou multicastu každých 5 sekúnd a obsahujú aj informácie o nakonfigurovaných parametroch, ako napríklad číslo procesu, alebo hodnoty K-parametrov. Špeciálnym typom Hello paketu je Ack, ktorý slúži na potvrdenie doručenia paketu nastavením potvrdzovacieho čísla na sekvenčné číslo prijatého paketu.

Update pakety sú používané na prenos informácií o cieľových sieťach a ich dostupnosti. Unicast update pakety sa posielajú v prípade objavenia nového suseda, alebo ak predchádzajúci multicast update paket nebol potvrdený a je potrebné ho preposlať. Na rozdiel od iných typov paketov, sú update posielané vždy spoľahlivo.

Pomocou **query paketov** smerovač oznamuje svoju aktuálnu vzdialenosť do siete, ktorá je v aktívnom stave, pretože žiadny zo susedov nespĺňa feasibility condition. Query pakety sú posielané spoľahlivým multicastom. Ak prijatý query paket obsahuje viaceré siete, DUAL musí každú z nich spracovať osobitne a ak je medzi nimi sieť, ktorá sa nenachádza v smerovacej tabuľke, smerovač na túto cestu odpovie reply paketom s nekonečnou metrikou.

Reply pakety sú vždy posielané spoľahlivým unicastom ako odpoveď na query pakety. V tele oznamujú aktuálnu vzdialenosť do cieľovej siete, pričom berú do úvahy aj zvýšenú vzdialenosť zistenú z príbuzných query paketov.

4.1.3 Reliable Transport Protocol

EIGRP využíva svoj vlastný transportný protokol s názvom Reliable Transport Protocol (RTP), ktorý bol viazaný patentom spoločnosti Cisco. Pre EIGRP zabezpečuje spoľahlivé doručenie paketov všetkým susedným smerovačom. RTP podporuje unicast aj multicast prenos paketov, pričom spoľahlivé doručenie je voliteľné, teda EIGRP môže

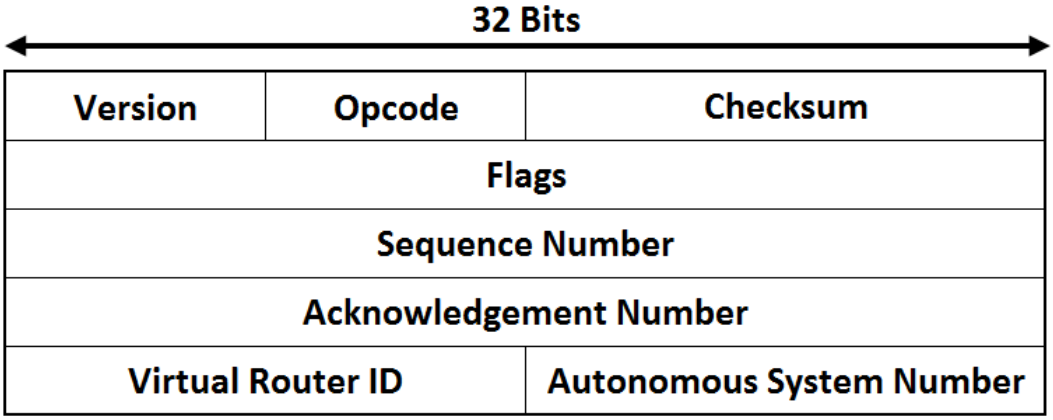
niektoré pakety poslať spoľahlivo a niektoré bez potvrdzovania. Algoritmus DUAL by bez RTP nemohol správne pracovať, pretože predpokladá, že komunikácia medzi susedmi je spoľahlivá a pakety sú doručované v správnom poradí. Režijné informácie, ako napríklad sekvenčené a potvrdzovacie číslo, sú zaznamenávané pre každého suseda osobitne.

Spoľahlivý prenos je zabezpečený pomocou sekvenčného a potvrdzovacieho čísla. Každý smerovač má svoje globálne sekvenčné číslo, ktoré vkladá do poľa sekvenčného čísla do paketov doručovaných spoľahlivo. Po odoslaní spoľahlivého paketu sa globálne sekvenčné číslo inkrementuje a pre každého suseda si smerovač poznačí, že požaduje potvrdenie. Ak smerovač dostane potvrdenie, značku zruší, ak nedostane, paket pošle znova. Ak ani po 16-tich pokusoch nedostane potvrdenie, relácia s prislúchajúcim susedom sa preruší. Pakety, ktoré nevyžadujú spoľahlivé doručenie, majú sekvenčné číslo nastavené na hodnotu 0 a pakety, ktoré nepotvrdzujú doručenie iného paketu majú v poli potvrdzovacieho čísla nastavenú 0.

Transportný protokol RTP štandardne nepoužíva viac ako 50% nakonfigurovanej kapacity rozhrania. Avšak ak nakonfigurovaná kapacita linky nezodpovedá reálnej kapacite, môže EIGRP, hlavne pri pomalých linkách, vygenerovať viac prevádzky než rozhranie zvládne preniesť, čo môže spôsobiť straty smerovacích paketov a nedostatočný výkon smerovacieho protokolu, alebo nedostatočnú kapacitu pre dáta používateľov.

4.1.4 Hlavička paketu

Každý EIGRP paket pozostáva z hlavičky, za ktorou nasleduje nula alebo viac polí s variabilnou dĺžkou nazývaných Type-Length-Value (TLV). Hlavičku EIGRP paketu je možné považovať aj za hlavičku RTP protokolu, keďže sú v nej obsiahnuté všetky polia potrebné na správne fungovanie tohto transportného protokolu.



Obrázok 6: Hlavička EIGRP paketu, zdroj autor

Dĺžka hlavičky v každom type paketu je 20 bajtov, pričom jej štruktúra je znázornená na obrázku (Obrázok 6).

Význam jednotlivých polí v hlavičke je nasledovný:

Version určuje verziu formátu EIGRP paketov, ktorá sa ale od uvedenia protokolu nezmenila a je nastavená na hodnotu dva.

Parameter Opcode určuje typ EIGRP správy, ktorá je reprezentovaná paketom. Hodnota 1 indikuje Update paket, 3 - Query, 4 - Reply, 5 - Hello a Ack.

Pole Checksum obsahuje kontrolnú sumu EIGRP paketu. Pole v hlavičke je pred výpočtom potrebné nastaviť na hodnotu nula. Samotný výpočet prebieha sčítaním celého paketu (hlavičky aj tela) po dvoch bajtoch. K súčtu sa následne vykoná jednotkový doplnok, ktorý sa zapíše do poľa checksum v hlavičke.

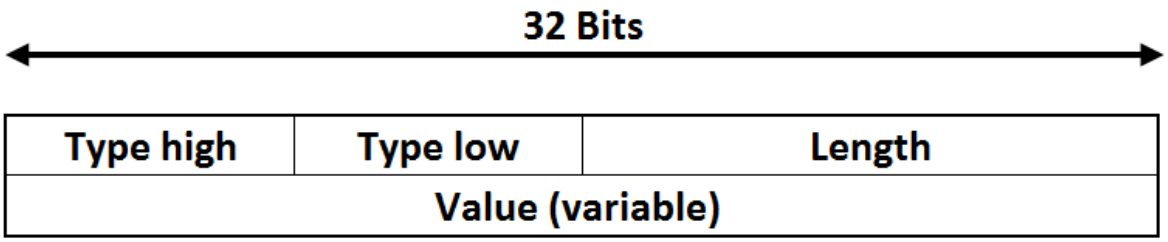
Nenulová hodnota v poli Flags indikuje potrebu narábať s paketom špeciálnym spôsobom. Príznak Init (hodnota 1) indikuje počiatočný stav susedstva a nutnosť úplnej synchronizácie topologických tabuliek. Conditional Receive, reprezentovaný hodnotou 2, hovorí smerovaču, že má byť spracovaný iba v prípade režimu Conditional Receive. Príznak Restart (hodnota 4) reprezentuje šetrný reštart smerovača, ktorý poslal paket a End-of-Table s hodnotou 8 indikuje ukončenie posielania obsahu topologickej tabuľky.

Polia Sequence number (sekvenčné číslo) a Acknowledgment number (potvrdzovacie číslo) boli popísané v kapitole 4.1.3.

Číslo Autonomous System Number musí byť na susedných smerovačoch rovnaké, pretože sa nepriamo používa aj pri autentifikácii. Ak sused prijme paket s číslom, ktoré sa nezhoduje s jeho vlastným, paket zahodí.

4.1.5 Formát Type-Length-Value

Každý EIGRP paket môže obsahovať variabilný počet TLV štruktúr. Prvou položkou TLV štruktúry je identifikátor typu, za ktorou nasleduje dĺžka celej TLV štruktúry, vrátane typu a dĺžky. Výhoda takého typu štruktúry spočíva v možnosti jednoducho preskočiť celé TLV, ktoré nie je podporované. Vďaka tomu je možná koexistencia starších a novších verzií EIGRP v jednej sieti. Všeobecná štruktúra TLV je znázornená na obrázku (Obrázok 7).



Obrázok 7: Formát TLV v EIGRP, zdroj autor

Vyšší bajt typu (Type high) definuje delenie TLV podľa protokolu, pre ktorý sa používa na: základné, IPv4, IPv6 a multiprotokolové. Nižší bajt typu (Type low) špecifikuje konkrétne TLV v rámci použitého protokolu. Zverejnený štandard EIGRP popisuje veľké množstvo TLV štruktúr, z ktorých uvedieme zopár najdôležitejších:

Parameter Type TLV slúži na prenos koeficientov na výpočet metriky (K-parametrov). Prenáša sa výlučne v Hello paketoch.

Authentication Type TLV sa využíva na prenos informácií o autentifikácii správ.

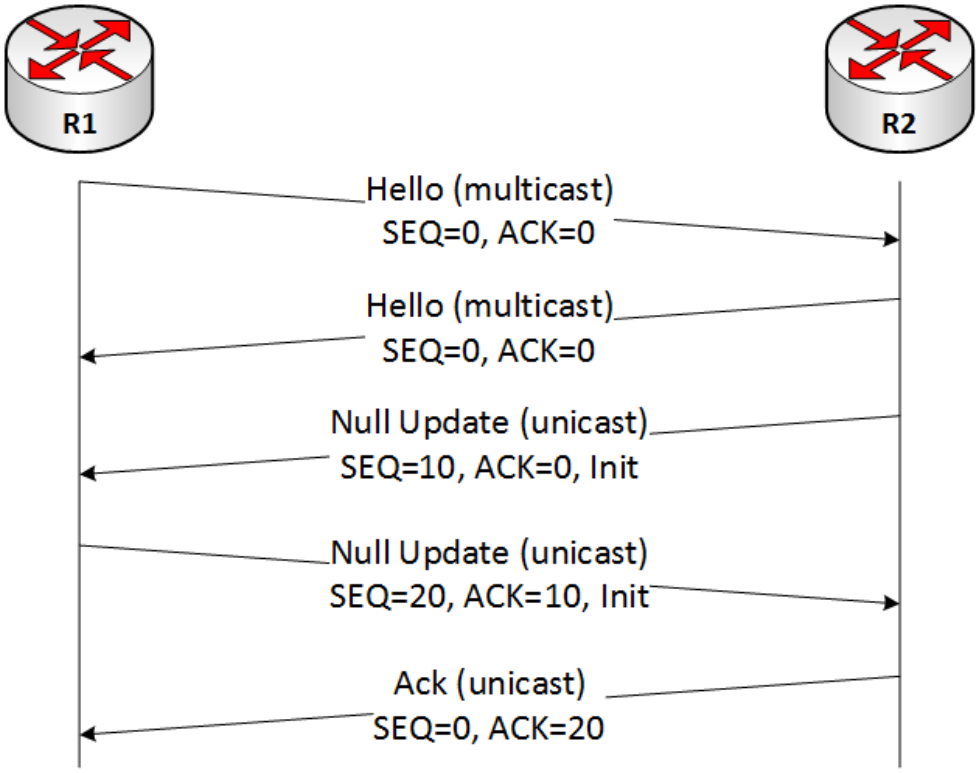
Software Version Type TLV indikuje implementovanú verziu EIGRP, a taktiež verziu formátu TLV štruktúr.

Na prenos smerovacích informácií spolu s prislúchajúcimi metrikami sa využívajú nasledovné TLV: IPv4 Internal Type, IPv4 External Type, IPv6 Internal Type a IPv6 External type.

4.1.6 Vytváranie susedských vzťahov

Ako už bolo spomenuté v kapitole 2, EIGRP si na rozdiel od ostatných smerovacích protokolov typu distance-vector udržiava susedské relácie. Smerovače objavia susedov v rámci broadcastovej domény, porovnajú konfiguračné parametre a kompatibilitu smerovacieho protokolu. Následne vytvoria susedstvo, oznamujú susedom, že sú stále k dispozícii a udržiavajú si informácie o stave a dostupnosti každého suseda. Objavenie suseda a udržiavanie susedského vzťahu je zabezpečované pomocou multicast Hello paketov.

Vybudovanie susedstva začína objavovaním susedov okamžite po aktivovaní EIGRP na smerovači. Kompletný proces je znázornený na obrázku (Obrázok 8).



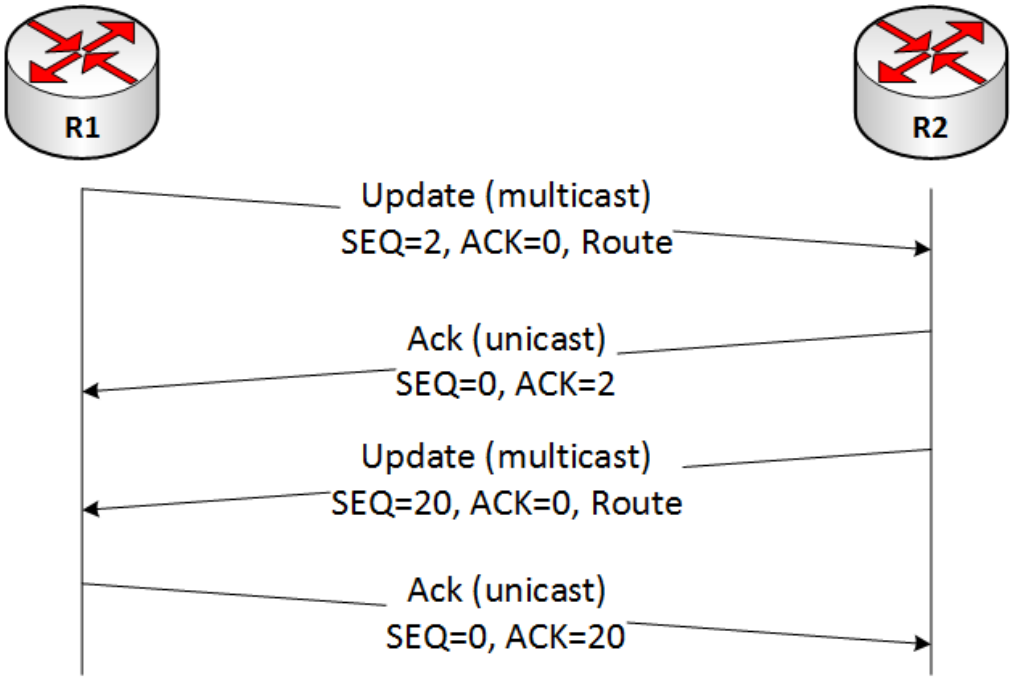
Obrázok 8: Vybudovanie susedstva v EIGRP, zdroj autor

Po objavení suseda si smerovače porovnajú identifikátor procesu a K-parametre potrebné pre výpočet metriky, ktoré sa nachádzajú v Hello paketoch. Smerovače sa stanú susedmi len v prípade, že sa ID procesu a K-parametre zhodujú. Novo objavený smerovač je pridaný do tabuľky susedov a jeho stav je nastavený na *Pending*. Susedovi v tomto stave neposielame ani neprijímame od neho žiadne pakety so smerovacími informáciami, a teda prípustné pakety v tejto fáze vytvorenia susedstva sú Update s príznakom Init a Ack pakety. Následne je potrebné overiť obojsmernú komunikáciu pomocou unicastu (multicast obojsmerná komunikácia bola už overená pomocou Hello paketov). Smerovač preto odošle unicast Null Update paket s nastaveným sekvenčným číslom a príznakom Init a čaká na potvrdenie doručenia. Tento paket je nazvaný ako Null Update preto, lebo neobsahuje smerovacie informácie a Init príznak znamená počiatočné vytvorenie susedstva, po ktorom je potrebné vykonať synchronizáciu kompletných smerovacích tabuliek, nie len poslať rozdielové aktualizácie. Sused taktiež pošle unicast Null Update paket s Init príznakom, ale nastaví svoje vlastné sekvenčné číslo a potvrdí prijatie predchádzajúceho paketu nastavením potvrdzovacieho čísla. Smerovač musí prijatý paket tiež potvrdiť, ale pretože už nie je potrebné posielať ďalší Update paket, na potvrdenie je využitý unicast Ack paket. Po vybudovaní susedstva smerovač zmení stav suseda na Up a smerovače si následne pomocou Update paketov začnú vymieňať smerovacie informácie.

Susedstvo je potrebné udržiavať periodickým posielaním multicast Hello paketov. Štandardná dĺžka intervalu posielania Hello paketov, ktorý je nazývaný ako Hello interval, je na Ethernet rozhraní 5 sekúnd. Ak smerovač neprijme od suseda žiadny Hello paket v priebehu určitého času (tzv. Hold Time intervalu), smerovač označí suseda ako nedostupného a susedstvo preruší. O zmenách je potrebné informovať aj DUAL, pretože rozpad susedstva môže ovplyvniť niektoré trasy do známych cieľových sietí. Hold Time interval je pri použití štandardnej konfigurácie nastavený na trojnásobok Hello intervalu, čo je v prípade Ethernet rozhrania 15 sekúnd. Na rozdiel od K-parametrov alebo identifikátoru procesu, nemusia byť časovače Hello interval a Hold Time na susedných smerovačoch nastavené na rovnakú hodnotu.

4.1.7 Výmena smerovacích informácií

Po ukončení procesu objavenia smerovača a vytvorenia susedstva je potrebné, aby si smerovače vymenili kompletné pracovné databázy. Proces výmeny smerovacích informácií medzi susedmi je znázornený na obrázku (Obrázok 9).



Obrázok 9: Výmena smerovacích informácií, zdroj autor

Smerovač (R1) pošle Update paket multicastom bez príznakov. Paket má nastavené sekvenčné číslo na hodnotu globálneho počítadla a potvrdzovacie číslo je nastavené na 0. V pakete sa nachádza rôzny počet TLV so smerovacími informáciami, podľa počtu položiek v smerovacej tabuľke. Následne susedný smerovač (R2) potvrdí prijatie predchádzajúceho Update paketu pomocou Hello paketu poslaným multicastom s nastaveným potvrdzovacím

číslom na sekvenčné číslo potvrdzovaného Update paketu. Potom sused pošle svoje vlastné smerovacie informácie rovnakým spôsobom ako R1. Keďže smerovač (R1) už nepotrebuje poslať ďalšie smerovacie informácie, potvrdí prijatie predchádzajúceho paketu ACK paketom, ktorý má nastavené potvrdzovacie číslo.

Akonáhle bola kompletná synchronizácia smerovacích tabuliek dokončená, susedia si posielajú len inkrementálne aktualizácie v prípade výskytu nejakej zmeny v sieti. Výnimku tvorí iba kompletný reštart smerovača alebo vynútená resynchronizácia smerovacích tabuliek.

4.1.8 Topologická tabuľka

Smerovací protokol EIGRP napĺňa topologickú tabuľku záznamami o vzdialených sieťach, ktoré mu oznámili susedné smerovače. Každej vzdialenej sieti prislúcha cieľová adresa, zoznam susedov, ktorí sieť ohlásili a metrika nazývaná Computed Distance (CD). CD je súčet vzdialenosti do cieľovej siete, pričom sa vypočítava sčítaním RD a ohodnotením rozhrania, ktorým sme pripojení ku susedovi. RD je metrika ohlásená susedom, pričom je to jeho vlastná vypočítaná CD.

EIGRP rozlišuje dva typy smerovacích záznamov v topologickej tabuľke: interné a externé. Interné cesty sa smerovač naučil od susedov v rámci jedného autonómneho systému a sú vždy preferované DUAL-om pri výbere successora. Externé smerovacie záznamy pochádzajú z iného zdroja, ako je napríklad iný AS, alebo boli redistribuované z iného smerovacieho protokolu.

4.1.9 Výpočet metriky

EIGRP používa dva typy vzorcov na výpočet metrík, ktoré sa nazývajú: klasické metriky (angl. *Classic Metric*) a široké metriky (angl. *Wide Metric*). Štýl výpočtu klasickej metriky bol prebratý zo smerovacieho protokolu IGRP. Ako súčasť výpočtu sa využívajú šírka pásma, oneskorenie, zaťaženie linky a spoľahlivosť. Vzťah na výpočet klasickej metriky je reprezentovaný rovnicou (Rovnica 1).

$$256 * \left\{ \left[\left(K1 * \frac{10^7}{BWmin} \right) + \frac{K2 * \frac{10^7}{BWmin}}{256 - zaťaženie} + \left(K3 * \sum oneskorenie \right) \right] * \frac{K5}{spoľahlivosť + K4} \right\}$$

Rovnica 1 : Klasická kompozitná metrika

BWmin reprezentuje najnižšiu šírku pásma rozhrania na celej trase do cieľovej siete v kbps. Význam parametrov, zaťaženie linky a spoľahlivosť sú zrejmé už z názvu, pričom sú vyjadrené ako percentuálna hodnota ale zapísané v hraniciach od 1 do 255. Suma oneskorení je súčet oneskorenia každého rozhrania po trase do cieľovej siete vyjadrený v desiatkach mikrosekúnd. Ak je oneskorenie nastavené na maximálnu hodnotu, sieť je nedostupná. Štandardne sú parametre K1 a K3 nastavené na hodnotu 1 a ostatné K-parametre na 0, čo zjednodušuje výpočet vzorca.

Klasická metrika nie je použiteľná ak sú rozhrania v topológii rýchlejšie ako 10Gbps, pretože ak máme v sieti napríklad rozhranie s rýchlosťou 10Gbps a iné s rýchlosťou 40Gbps, časť vzorca klasickej metriky berúca do úvahy rýchlosť linky ($10\text{^}7/BWmin$) bude mať v oboch prípadoch rovnakú hodnotu. Pre 10Gbps: $(10^7)/(10^7) = 1$ a pre 40Gbps: $(10^7)/4 * (10^7) = 0{,}25$. Keďže EIGRP pri výpočte metriky využíva iba celé čísla, hodnota 0,25 bude tiež reprezentovaná číslom 1.

Aby bolo možné rozlíšiť aj rýchlejšie rozhrania, klasická metrika bola prepísaná na tvar reprezentovaný vzťahom (Rovnica 2).

$$\left\{\left[\left(K1 * \frac{10^7 * 65536}{BWmin}\right) + \frac{K2 * \frac{10^7 * 65536}{BWmin}}{256 - zaťaženie} + \left(K3 * \sum \frac{oneskorenie * 65536}{10^6}\right) + (K6 * ExtAttr)\right] * \frac{K5}{spoľahlivosť + K4}\right\}$$

Rovnica 2 : Široká kompozitná metrika

Parametre BWmin a oneskorenie boli naškálované aby vyhovovali aj rýchlejším rozhraniam. Pribudol aj nový parameter ExtAttr, ktorý reprezentuje jitter a spotrebu energie. Rovnako ako pri klasickej metrike, aj široká kompozitná metrika má štandardne K1 a K3 koeficienty nastavené na 1 a ostatné na hodnotu 0.

4.1.10 Split Horizon a Poisoned Reverse

Už základný dizajn EIGRP protokolu založený na FC zabraňuje vznikaniu smerovacích slučiek aj bez iných pomocných mechanizmov. Avšak pre zvýšenie efektívnosti výmeny smerovacích informácií sú v EIGRP implementované mechanizmy Split Horizon a Split Horizon s Poisoned Reverse.

Mechanizmus Split Horizon slúži na potlačenie šírenia query a update paketov. V prípade jeho použitia sa sieť nikdy neohlasuje cez rozhranie, ktoré slúži ako next hop do tejto cieľovej siete. Striktnejšou formou tohto mechanizmu je jeho modifikácia s Poisoned

Reverse, ktorá sieť ohlási ako nedostupnú cez next hop rozhranie do tejto konkrétnej cieľovej siete.

4.1.11 Nekompletné alebo chýbajúce funkcie

Aj keď RFC 7868 obsahuje popis všetkých vlastností a funkcií potrebných na základnú implementáciu protokolu, popis niektorých pokročilých funkcií je buď nekompletný, alebo úplne chýba. Napríklad RFC obsahuje formát a základný popis TLV použitého na autentifikáciu, ale nepopisuje priebeh autentifikačného mechanizmu ani postup, ako správne vypočítať autentifikačné hash-e. Príkladom kompletne chýbajúceho popisu funkcionality je filtrácia smerovacích informácií alebo funkcia EIGRP Stub.

4.2 Otvorené implementácie EIGRP

V kapitole 2 sme spomenuli, že uvoľnenie špecifikácie protokolu EIGRP v roku 2013 umožnilo vznik rôznych open-source implementácií tohto pôvodne proprietárneho protokolu. V tejto podkapitole sa budeme venovať každej z dostupných implementácií, priblížime ich špecifiká a porovnáme dostupnú funkcionalitu voči RFC a referenčnej Cisco implementácii smerovacieho protokolu.

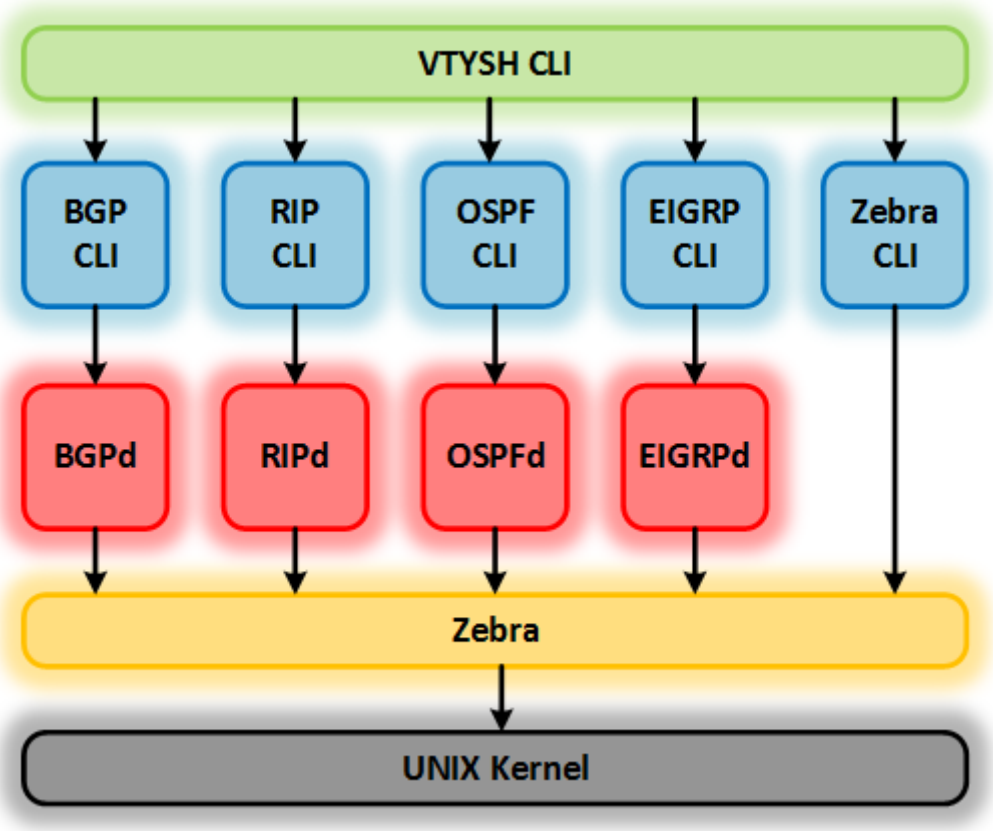
4.2.1 Quagga

Quagga je otvorený softvérový balík smerovacích protokolov so zdrojovými kódmi dostupnými pod licenciou GPLv2. Dôsledkom použitia licencie GPLv2 je, že zdrojové kódy je dovolené bezplatne používať, distribuovať, ako aj preskúmavať a modifikovať za predpokladu, že sú ďalej šírené a uvoľnené pod rovnakou licenciou. Quagga vznikla v roku 2002 ako fork projektu GNU Zebra, ktorý už v tom období obsahoval implementáciu protokolov OSPFv2, RIP a BGP. Keďže sa vývoj zebry spomaľoval a opravy prichádzali po dlhom čase, väčšina komunity rýchlo adaptovala nový projekt Quagga. V súčasnosti Quagga ponúka nielen podporu ďalších protokolov, ako sú OSPFv3, IS-IS, OIM, Babel, LDP, ale aj rôznych rozšírení protokolov BGP a OSPF. Niektoré protokoly, ako napríklad EIGRP, sú dostupné iba vo vývojových vetvách a neboli začlenené do stabilných verzií softvérového balíka. Quagga je napísaná v jazyku C a je ju možné používať na rôznych operačných systémoch ako Linux, OpenBSD, FreeBSD, Solaris, alebo aj na vstavaných zariadeniach ako sú napríklad smerovače s operačným systémom OpenWRT [44]–[46].

Architektúra Quaggy sa výrazne odlišuje od ostatných smerovacích softvérov, keďže pozostáva z viacerých procesov, nazývaných aj ako démony, ktoré spolu pracujú na napĺňaní

smerovacej tabuľky záznamami. Schéma prepojenia jednotlivých modulov v Quagge je zobrazená na obrázku (Obrázok 10).

Hlavným modulom Quaggy je démon Zebra, ktorého názov je odvodený od pôvodného projektu GNU Zebra. Slúži ako prepojenie medzi jadrom operačného systému a ostatných modulov, ktoré sú súčasťou Quaggy. Spravuje sieťové rozhrania, smerovaciu tabuľku, ako aj redistribúciu smerovacích záznamov medzi rôznymi smerovacími protokolmi.



Obrázok 10: Schéma architektúry Quaggy, zdroj autor

Každý smerovací alebo iný sieťový protokol je reprezentovaný samostatným modulom, ktorý vykonáva iba základné funkcie spojené s konkrétnym protokolom. Funkcie, ktoré sú rovnaké pre všetky smerovacie protokoly, akými sú zistenie stavu rozhrania, poslanie alebo prijatie paketu alebo vloženie naučeného smerovacieho záznamu do smerovacej tabuľky, sú zabezpečované modulom Zebra. Preto je nevyhnutné, aby každý z týchto modulov vedel správne komunikovať so Zebrou. Výhodou popísanej modulárnej architektúry je jednoduchšia správa a oprava chýb a nižšie nároky na prostriedky smerovača (je možné aktivovať iba potrebné moduly, nie len všetky dostupné), ako aj pridanie úplne

nového sieťového protokolu (stačí vytvoriť nový démon a implementovať komunikáciu so Zebrou) [44], [45].

Jednou z najbadateľnejších nevýhod modulárnej architektúry Quaggy je samostatný konfiguračný súbor a príkazový riadok (angl. *Command-line Interface - CLI*) slúžiaci na interakciu s démonom. Napríklad v prípade aj jednoduchej konfigurácie akou je pridanie statickej default route a následné zapnutie smerovacieho protokolu EIGRP na rozhraní, je potrebné interagovať s CLI modulu Zebra ako aj modulu EIGRP. Súčasť Quaggy, ktorá rieši tento konkrétny problém, je spoločné terminálové rozhranie VTYSH, ktoré prepája CLI každého modulu a umožňuje vykonať následnú konfiguráciu viacerých modulov na jednom mieste [44], [46].

Syntax jednotlivých terminálových príkazov v Quagge nápadne pripomína príkazy smerovačov s operačným systémom Cisco IOS, čo prispieva k rýchlemu osvojeniu si konfigurácie a administrácie riešenia postaveného na Quagge administrátormi sietí [45].

Otvorená implementácia smerovacieho protokolu EIGRP do Quaggy sa zrodila v roku 2014 na Katedre informačných sietí Žilinskej univerzity, pod vedením Ing. Petra Palúcha, PhD ako súčasť projektovej výučby. Do projektu sa zapojilo viacero študentov a ich práca bola oficiálne uznaná aj v rámci RFC 7868 [41]. Vo svojej práci pokračovali aj počas vypracovania diplomovej práce. V roku 2015 do tímu pribudli ďalší študenti, ktorí implementáciu rozšírili a ich práca bola taktiež pretavená do diplomových prác.

Základná funkcionalita, dokončená v roku 2015, bola rozdelená na tri celky: transportnú časť, riadiacu časť a DUAL. Aby bola spolupráca na projekte čo najefektívnejšia, zdrojové kódy boli umiestnené na GitHub-e [47] a ako vývojové prostredie bolo použité Eclipse IDE, ktoré uľahčuje prácu so zdrojovými kódmi a ladením projektu [48]–[50].

Transportná časť implementácie začala po dôkladnom naštudovaní štruktúry smerovacích démonov v Quagge vytvorením samotného EIGRP modulu. Následne bolo vytvorené prepojenie so Zebrou, ktoré umožnilo získať informácie o rozhraniach a položkách smerovacej tabuľky. Avšak jedna z najdôležitejších funkcionalít transportnej časti bola implementácia RTP protokolu podľa špecifikácie popísanej v kapitole 4.1.3. Po dokončení RTP implementácie začala práca na vytváraní a správe susedských relácií, ako aj periodické posielanie Hello paketov. Následne bola implementovaná výmena smerovacích informácií pomocou interných aj externých TLV, teda aj s podporou redistribúcie. Aj

napriek nedostatočnému popisu v RFC, bola po niekoľkých testoch v rámci topológie s Cisco smerovačmi implementovaná aj podpora autentifikácie. Súčasťou tejto časti bolo aj ukladanie aktuálnej konfigurácie EIGRP modulu do konfiguračného súboru [48].

Časť riadenia začala implementáciou konzolových príkazov, pomocou ktorých prebieha samotná konfigurácia EIGRP procesu, ako napríklad zapnutie smerovacieho protokolu na rozhraní alebo povolenie autentifikácie správ. Následne boli zahrnuté príkazy na zobrazenie stavu EIGRP procesu a ladiace príkazy a výpisy. Časť riadenia bola ukončená implementáciou podpory SNMP protokolu na získavanie stavových informácií o smerovacom procese [50].

Časť zameraná na algoritmus DUAL začala implementáciou metriky na virtuálnych rozhraniach používaných na výpočet najkratších ciest v EIGRP module. Následne boli vytvorené všetky štruktúry a obslužné funkcie potrebné na správu topologickej tabuľky. Samotná topologická tabuľka bola implementovaná pomocou údajovej štruktúry lineárne zreťazený zoznam. V tejto časti bola implementovaná aj podpora TLV potrebných pre klasické a široké metriky. Zvyšok časti sa venuje implementácii stavového automatu DUAL podľa špecifikácie v RFC [49].

V roku 2016 bola základná implementácia rozšírená o funkcionalitu, ktorá v RFC nie je popísaná: filtrácia smerovacích záznamov [51] a mechanizmy pre reštart susedských relácií [46].

Implementácia filtrácie smerovacích záznamov bola vytvorená na základe správania referenčného EIGRP na smerovačoch od spoločnosti Cisco. Viaceré zo základných príkazov potrebných na konfiguráciu filtrácie už boli v Quagge prítomné, keďže sa používajú aj v iných smerovacích protokoloch. Samotná filtrácia v EIGRP bola vytvorená pre prichádzajúce aj odchádzajúce smerovacie informácie, pričom konfiguračné pravidlá môžu byť vytvorené pomocou nástrojov ako sú access-list, prefix-list a distribute-list. Filtrovanie pomocou nástroja route map dokončené nebolo [51].

Aby bolo možné správne pochopiť a implementovať podporu mechanizmov pre reštart susedských vzťahov, bolo potrebné pozorovať komunikáciu medzi viacerými smerovačmi s referenčnou Cisco EIGRP implementáciou. Nevyhnutné tiež bolo vytvoriť nástroj, ktorý umožnil skonštruovať a poslať umelý EIGRP paket (kombinácia rôznych príznakov a typov paketov, po prípade TLV štruktúr) a pozorovať reakcie Cisco smerovačov na tieto pakety. Na základe týchto pokusov a pozorovaní bola implementovaná podpora

tvrdého aj mäkkého reštartu, napríklad po aplikácii mechanizmov na filtráciu smerovacích informácií [46].

Aj keď implementácia EIGRP v softvérovom smerovacom balíku Quagga obsahuje všetky základné funkcie potrebné na prevádzku EIGRP v IPv4 topológii, podpora pre sieťový protokol IPv6 chýba, čo bráni jej nasadeniu v mnohých súčasných produkčných sieťach. Podpora EIGRP sa taktiež doteraz nedostala do stabilných vydaní Quaggy a je dostupná iba vo vývojovom repozitári na GitHube [47].

4.2.2 Free Range Routing

Free Range Routing (FRR) taktiež známy aj ako FRRouting vznikol v roku 2017 ako fork Quaggy vývojármi a spoločnosťami, ktoré neboli spokojné so spôsobom prispievania do projektu Quagga a celkovo pomalým začleňovaním opráv a nových funkcií [52]. FRR je momentálne pod záštitou organizácie Linux Foundation [53] a zdrojové kódy celého projektu sú dostupné na GitHube [54], čo uľahčuje a zrýchľuje proces pridávania nových funkcií a opráv, ako aj overenie týchto zmien hlavnými správcami projektu.

Po vzniku FRR bol zdrojový kód implementácie EIGRP do Quaggy začlenený priamo do vývojovej vetvy FRR. Odvtedy vývojári FRR dôkladne skontrolovali EIGRP implementáciu a opravili viacero chýb v kóde, alebo optimalizovali niektoré časti kódu pre zlepšenie výkonu. Ako príklad opravy chyby môžeme uviesť situáciu, keď implementácia vytvorila paket väčší ako MTU rozhrania a pokúsila sa ho poslať, čo viedlo k tomu, že celý smerovací proces spadol. Príkladom optimalizácie časti kódu je topologická tabuľka, ktorá bola pôvodne reprezentovaná údajovou štruktúrou lineárne zreťazený zoznam a bola nahradená štruktúrou tabuľka [54].

Doteraz ešte neboli pridané žiadne nové funkcie, takže ani vo FRR EIGRP nepodporuje IPv6 protokol a jediným zoznamom zmien sú commity v repozitári na GitHube. Avšak v marci 2018 bola uvoľnená prvá stabilná verzia FRR (4.0), ktorá obsahuje EIGRP démon [54].

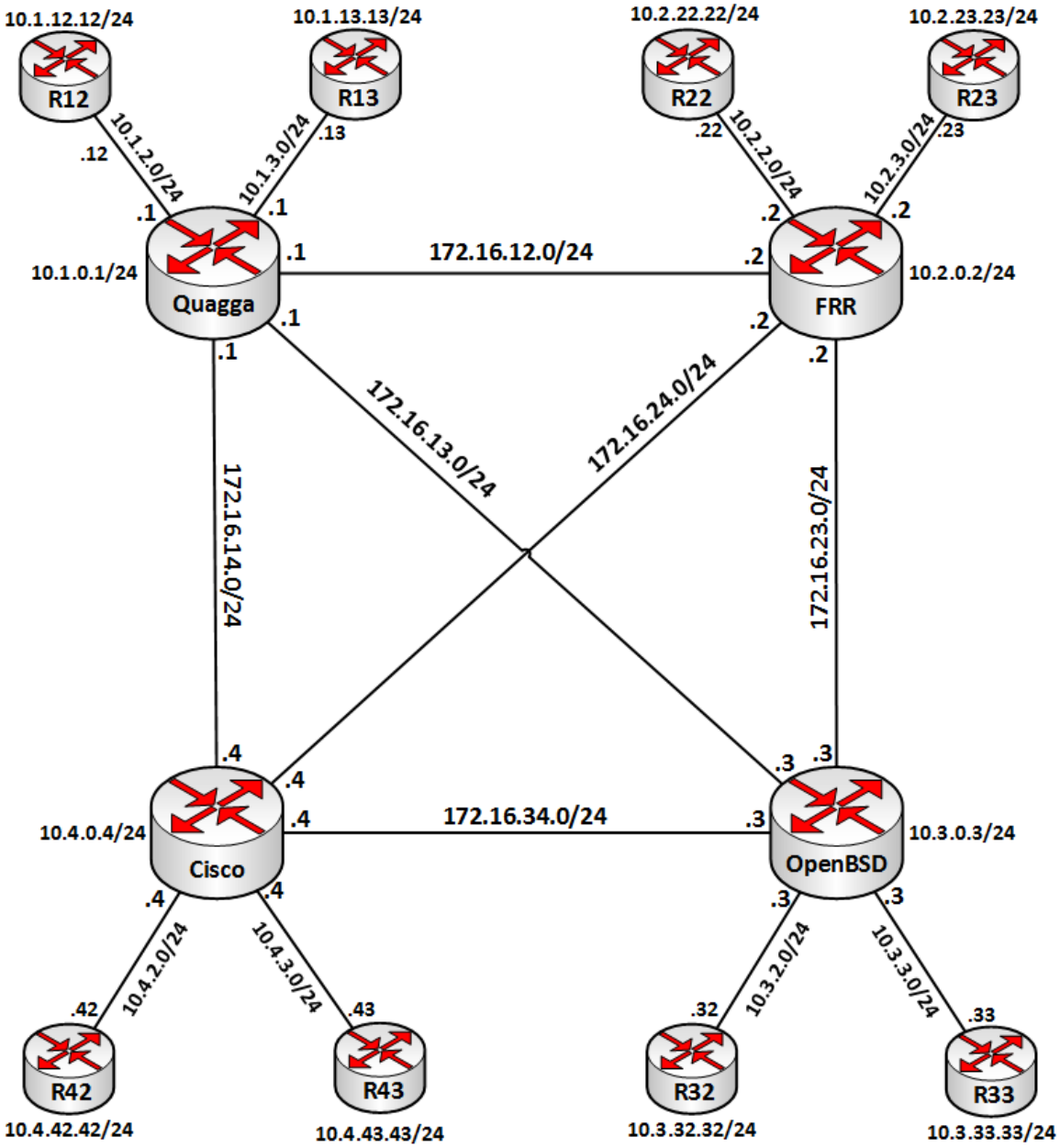
4.2.3 OpenBSD

Implementácia smerovacieho protokolu EIGRP pre operačný systém OpenBSD bola vyvíjaná v rokoch 2015 až 2016 vývojárom Renato Westphal, ktorý v súčasnosti pracuje na projekte FRR. Implementácia pozostáva zo samostatného programu s názvom „eigrpd“, ktorý zabezpečuje všetku funkcionalitu EIGRP protokolu, ako aj komunikáciu s jadrom operačného systému za účelom manipulácie so smerovacou tabuľkou a sieťovými

rozhraniami. Druhý program s názvom „eigrpctl“ slúži na zistenie a zobrazenie stavu programu eigrpd. Zmena nastavení smerovacieho procesu je vykonávaná úpravou konfiguračného súboru „/etc/eigrpd.conf“ a následným reštartom celého démonu eigrpd. Všetky funkcie tejto implementácie boli vyvíjané na základe RFC špecifikácie, pričom sú k dispozícii všetky základné prvky ako DUAL, RTP ale aj Stuck in Active (SIA) a režim Conditional Receive (CR), ale aj pokročilé prvky, ako sumarizácia a podpora viacerých procesov. Na rozdiel od implementácie EIGRP, v Quagge a FRR sú podporované oba v súčasnosti používané sieťové protokoly, IPv4 aj IPv6. Avšak podpora funkcií, ktoré nie sú v RFC popísané, chýba úplne. Príkladom je autentifikácia alebo podpora filtrovania smerovacích informácií. Zdrojové kódy sú dostupné na GitHube [55].

4.3 Testovanie funkcií otvorených implementácií

Keďže nie všetky otvorené implementácie podporujú všetky funkcie popísané v RFC 7868 a dostupné v referenčnej implementácií od spoločnosti Cisco, ako sme spomenuli v kapitole 4.2, bolo potrebné otestovať dostupné funkcie. Na otestovanie sme vytvorili topológiu smerovačov zobrazenú na obrázku (Obrázok 11).



Obrázok 11: Testovacia topológia otvorených implementácií EIGRP, zdroj autor

Pre zjednodušenie a zrýchlenie testovania bola topológia spustená v simulačnom nástroji GNS3, ktorý umožňuje spúšťať rôzne virtualizované smerovače a operačné systémy

za pomoci externých nástrojov. Všetky smerovače so systémom Cisco IOS (Cisco, R12, R13 .... R42, R43) boli emulované pomocou nástroja Dynamips. Smerovače Quagga a FRR boli vytvorené pomocou kontajnerového nástroja Docker, pričom ako základný systém bola použitá distribúcia Linuxu Ubuntu 18.04. Keďže operačný systém OpenBSD vo verzii 6.3 nebolo možné spustiť v nástroji Docker, na virtualizáciu bol použitý nástroj VirtualBox.

Štyri hlavné smerovače (Quagga – R1, FRR - R2, OpenBSD – R3, Cisco – R4) sú prepojené Ethernet linkami každý s každým, pričom linky sú adresované sieťami v tvare 172.16.XX.0/24, kde XX sú čísla prepojených smerovačov vzostupne a host časť adresy vždy zodpovedá číslu smerovača. Preto na prepoji Quagga – FRR má Quagga adresu 172.16.12.1 a FRR adresu 172.16.12.2. Každý z týchto štyroch smerovačov má priradenú aj loopback adresu v tvare 10.X.0.X/24, kde X je opäť poradové číslo smerovača.

Smerovací protokol EIGRP je spustený na linkách medzi štyrmi hlavnými smerovači, pričom na každom z nich beží ešte protokol OSPF, z ktorého sa cesty redistribuujú do EIGRP. Smerovací protokol OSPF je tiež spustený na smerovačoch R12 – R42 s operačným systémom Cisco IOS, ktoré sú vždy po dvojici pripojené k jednému zo štyroch hlavných smerovačov. Linky medzi hlavným smerovačom a vedľajšími sú adresované v tvare 10.X.Y.0/24, kde X je poradové číslo hlavného smerovača a Y vedľajšieho smerovača. Preto v prepoji FRR – R23 má FRR adresu 10.2.3.2 a R23 adresu 10.2.3.23. Každý z vedľajších smerovačov má pridelenú aj loopback adresu v tvare 10.X.Z.Z/24, kde X je číslo hlavného smerovača a Z je číslo vedľajšieho smerovača. Preto má smerovač R22 loopback adresu 10.2.22.22/24.

Na testy EIGRP s IPv6 protokolom bola použitá iba časť topológie na obrázku (Obrázok 11), pozostávajúca z hlavných smerovačov Cisco a OpenBSD, spolu s prepojom medzi nimi a vedľajších smerovačov R32, R33, R42, R43. Dôvodom použitia iba časti topológie bolo, že na smerovačoch Quagga a FRR nebolo možné spustiť EIGRP pre IPv6, čo vyplýva aj z ich popisu v kapitolách 4.2.1 a 4.2.2.

Každý z podporovaných IPv6 smerovačov mal pridelenú statickú IPv6 adresu pre Ethernet aj loopback rozhranie odvodenú od použitej IPv4 adresy, preto napríklad smerovač R22 mal loopback IPv6 adresu nastavenú na fd00:10:2:22::22/64 a na prepoji OpenBSD – R32 mal OpenBSD adresu fd00:10:3:2::3/64.

Výsledky testovania sú zhrnuté v tabuľke (Tabuľka 2).

	Quagga	FRR	OpenBSD	
	IPv4	IPv4	IPv4	IPv6
nadviazanie susedstva	✓	✓	✓	✓
výmena smerovacích informácií	✓	✓	✓	✓
autentifikácia	✓	✓	✗	✗
filtrácia	✓	✓	✗	✗
sumarizácia	✓	✓	✓	✓
redistribúcia	✓	✓	✓	✓

Tabuľka 2: Výsledky testovania otvorených implementácií EIGRP

Počas testovania sme sa stretli s občasnými pádmi smerovacieho modulu EIGRP v balíku Quagga. Narazili sme taktiež na problém veľkého počtu smerovacích informácií, ktoré presahuje MTU rozhrania popísaný v kapitole 4.2.2. Výsledky testovania zodpovedajú stavu implementácií popísanými v kapitole 4.2.

5 Ciele práce

Ako sme spomenuli v úvode, v súčasnosti čelia smerovacie protokoly výzvam v podobe objavovania sa nových technológií ako aj ich použitia v nových oblastiach, ktoré sa odlišujú od prostredia klasických počítačových sietí. Nás zaujal hlavne problém využitia smerovacích protokolov v prostredí využívajúcom viaceré sieťové protokoly s vlastným adresovým systémom. Ako bolo zdôraznené v tretej kapitole, problematika činnosti smerovacích protokolov spolu s riešeniami ich implementácií je veľmi obsiahla. Preto sme sa rozhodli vykonať dekompozíciu funkcionalít smerovacieho protokolu s bližším zameraním len na parciálne časti priamo súvisiace s oblasťou činnosti v multiprotokolovom nasadení (a nie napr. aj problematiky behu smerovacích algoritmov, ktorá je nezávislá od prevádzkovaného protokolu). Za týmto účelom sa zameriame na analýzu častí/služieb smerovacieho protokolu, ktoré sú potrebné pre transport smerovacích informácií a správu susedstiev v komunikačnom prostredí používajúcom viaceré protokolové adresové rodiny.

Aby bolo možné rigorózne porovnať spôsoby riešenia transportu a susedstiev medzi súčasnými smerovacími protokolmi a navrhnúť zlepšenia, vykonali sme analýzu dostupnej literatúry. Na základe analýzy sa ako vhodný nástroj na vytvorenie modelu činnosti a porovnania jednotlivých služieb javia formálne metódy. Špecificky pre smerovacie protokoly sú vhodným typom formálnych metód farbené Petriho siete.

Na základe našich skúseností s otvorenou implementáciou smerovacieho protokolu EIGRP padla voľba práve na tento protokol ako vzor pre model tretej triedy protokolovej integrovanosti vytvorenej pomocou farbených Petriho sietí.

Pri modelovaní jednotlivých tried protokolovej integrovanosti je potrebné zakomponovať jednotlivé služby smerovacích protokolov nevyhnutné na prenos smerovacích informácií a správu susedstiev v prostredí viacerých adresových rodín. Namodelované triedy za pomoci farbených Petriho sietí je potrebné porovnať za účelom návrhu zlepšenia týchto procesov pre súčasné a budúce integrované smerovacie protokoly. Ako finálny výstup je potrebné formulovať odporúčania na základe výsledkov porovnania pre dizajnérov budúcich smerovacích protokolov.

Jednotlivé čiastkové ciele sú teda nasledovné:

- vytvorenie modelov jednotlivých tried protokolovej integrovanosti
- popísanie jednotlivých modelov

- analýza a porovnanie vytvorených modelov
- vytvorenie odporúčaní pre dizajn integrovaných smerovacích protokolov
- porovnanie odporúčaní s existujúcimi smerovacími protokolmi
- zhodnotenie.

6 Modelovanie tried riešenia protokolovej integrovanosti

Keďže cieľom práce bolo navrhnúť metodiku pre dizajn integrovaných smerovacích protokolov so zameraním na susedstvá a transport údajov smerovacieho protokolu, v kapitole 2 boli popísané štyri identifikované triedy kombinácií počtu susedstiev a spôsobu transportu informácií smerovacieho protokolu.

Aby bolo možné tieto štyri triedy navzájom porovnať a určiť, ktorá z tried je najvhodnejšia pre dizajn budúcich smerovacích protokolov, na základe našej analýzy literatúry, popísanej v kapitole 3, sme sa rozhodli vytvoriť modely každej zo štyroch tried pomocou farbených Petriho sietí za pomoci nástroja CPN Tools.

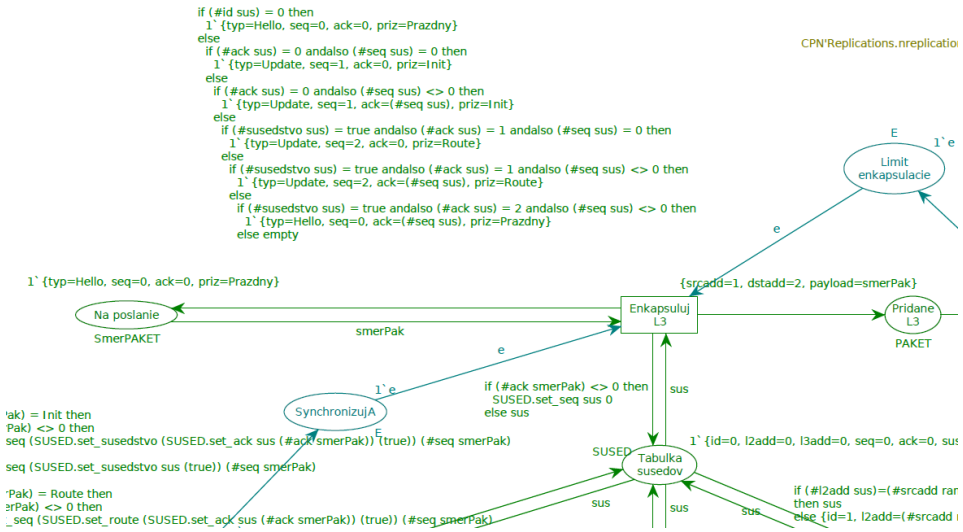
Keďže najviac znalostí z vnútornej štruktúry smerovacieho protokolu sme získali analýzou protokolu EIGRP, popísanou v kapitole 4, rozhodli sme sa ako prvú modelovať tretiu triedu, do ktorej tento protokol zaraďujeme a ako vzor modelu sme použili práve protokol EIGRP. Každý z modelov modeluje komunikáciu a stav práve dvoch smerovačov.

6.1 Tretia trieda – transport L3, viaceré susedstvá

Celá grafická reprezentácia modelu tretej triedy riešenia protokolovej integrovanosti, ktorá pozostáva z vytvorenia susedstiev pre oba sieťové protokoly, pričom správy smerovacieho protokolu sú prenášané na sieťovej vrstve, je zobrazená v prílohách F (časť reprezentujúca sieť a smerovač A) a G (časť reprezentujúca sieť a smerovač B). V nasledujúcich podkapitolách vysvetlíme funkcionalitu a dizajn jednotlivých prvkov modelu, pričom časti sú popísané v poradí, v akom bol model vytváraný.

6.1.1 Odosielacia časť

Vytváranie modelu tretej triedy protokolovej integrovanosti sme začali dizajnom častí protokolu nevyhnutných pre vytvorenie susedského vzťahu z pohľadu jedného smerovača (smerovač A). Ako prvé sme na základe procesu vytvorenia susedského vzťahu, popísaného v kapitole 4.1.6, vytvorili prvok generujúci správy smerovacieho protokolu.



Obrázok 12: Trieda 3 - Generovanie správ, tabuľka susedov

Prvok, ktorý generuje správy smerovacieho protokolu, je na obrázku (Obrázok 12) reprezentovaný miestom s názvom *Na poslanie*. Do tohto miesta môžu byť uložené značky s farbou *SmerPAKET*, čo je údajová štruktúra, definovaná v jazyku CPN ML ako:

```
colset SmerPAKET = record typ:TYP * seq:SEQ * ack:SEQ * priz:PRIZNAK;
```

Táto údajová štruktúra reprezentuje hlavičku smerovacieho protokolu EIGRP, popísanú v kapitole 4.1.4, ktorá ale bola zjednodušená a obsahuje iba nasledovné polia:

- typ – identifikuje Hello alebo Update paket
- seq – sekvenčné číslo potvrdzovaného paketu
- ack – potvrdzovacie číslo (sused potvrdzuje prijatie paketu)
- priz – pole príznakov (Prazdne, Init – vytvorenie susedstva, Route – prenos smerovacej informácie)

Miesto *Na poslanie* v úvode obsahuje jedinú správu, Hello paket, ktorý slúži na objavenie suseda a je definovaný nasledovne:

```
1`{typ=Hello, seq=0, ack=0, priz=Prazdny}
```

Ďalšie pakety potrebné na vytvorenie susedstva a prenos smerovacích informácií sú generované na základe informácií z tabuľky susedov, ktorá zachytáva stav, v akom sa susedstvo nachádza. Je reprezentovaná miestom *Tabulka susedov* pričom jednotlivé záznamy sú reprezentované značkou farby SUSED definovanej ako:

```
colset SUSED = record id:INT * l2add:INT * l3add:INT * seq:INT * ack:INT *
susedstvo:BOOL * route:BOOL;
```

Toto miesto, aj napriek názvu nezodpovedá tabuľke susedov v smerovacom protokole EIGRP, ale zachytáva stav, v akom sa susedstvo nachádza, ako aj informáciu o tom, či boli smerovacie záznamy prenesené. Môžeme ho považovať za veľmi zjednodušenú formu kombinácie niektorých funkcionalít v EIGRP zabezpečovaných algoritmom DUAL (4.1.1), tabuľkou susedov a topologickou tabuľkou (4.1.8). Položky, ktoré obsahuje každý záznam tabuľky susedov sú nasledovné:

- id – interný jednoznačný identifikátor suseda
- l2add – linková adresa suseda
- l3add – sieťová adresa suseda
- seq – správu s akým sekvenčným číslom treba susedovi potvrdiť
- ack – správu s akým sekvenčným číslom nám sused naposledy potvrdil
- susedstvo – informácia, či sme prijali správu s príznakom Init, čo značí, že sa začína vytvárať susedstvo
- route – informácia, či sme prijali správu so smerovacími záznamami.

Generovanie ďalších paketov do miesta Na poslanie, prebieha volaním nasledujúcej funkcie vždy, keď sa aktivuje prechod Enkapsuluj L3:

```
if (#id sus) = 0 then
  1`{typ=Hello, seq=0, ack=0, priz=Prazdny}
else
  if (#ack sus) = 0 andalso (#seq sus) = 0 then
    1`{typ=Update, seq=1, ack=0, priz=Init}
  else
    if (#ack sus) = 0 andalso (#seq sus) <> 0 then
      1`{typ=Update, seq=1, ack=(#seq sus), priz=Init}
    else
      if (#susedstvo sus) = true andalso (#ack sus) = 1 andalso (#seq sus)
= 0 then
        1`{typ=Update, seq=2, ack=0, priz=Route}
      else
        if (#susedstvo sus) = true andalso (#ack sus) = 1 andalso (#seq sus)
<> 0 then
          1`{typ=Update, seq=2, ack=(#seq sus), priz=Route}
        else
          if (#susedstvo sus) = true andalso (#ack sus) = 2 andalso (#seq
sus) <> 0 then
            1`{typ=Hello, seq=0, ack=(#seq sus), priz=Prazdny}
          else empty
```

Ako môžeme vidieť zo zdrojového kódu funkcie, na začiatku, keď smerovač ešte neprijal žiadnu Hello správu od suseda (id v tabuľke susedov je nastavené na hodnotu 0), sú generované len úvodné Hello správy.

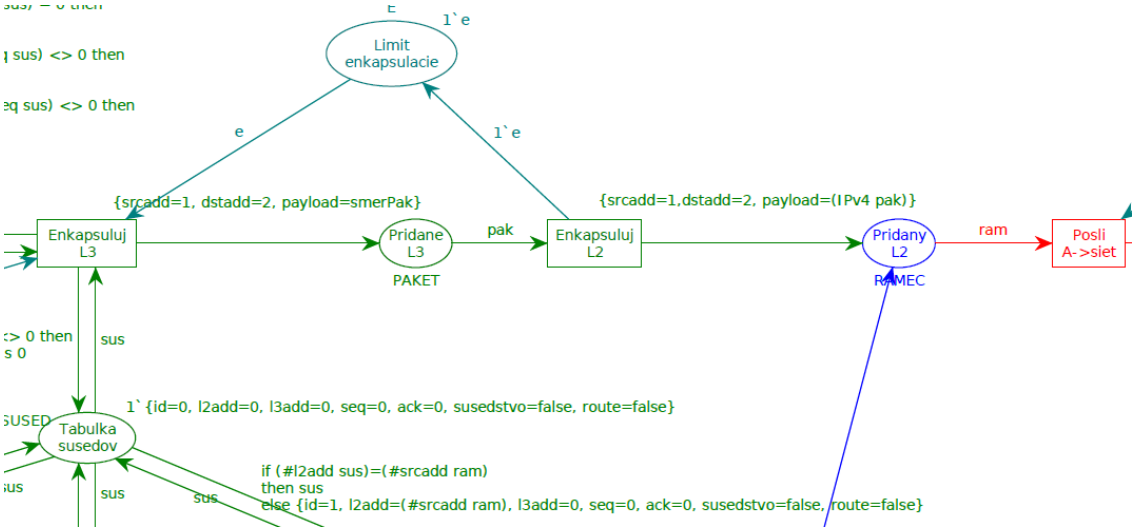
Akonáhle sme suseda objavili (prijali sme od neho Hello správu), id suseda je nastavené na nenulovú hodnotu, funkcia generuje Update správu so sekvenčným číslom 1 a Init príznakom. Potvrdzovacie číslo v tejto správe je nastavené na 0 ak netreba susedovi potvrdiť prijatie predchádzajúcej správy, ak treba, potvrdzovacie číslo je nastavené na sekvenčné číslo z tabuľky susedov.

Potom, ako je susedstvo vytvorené (prijali sme Update správu s Init príznakom a sused potvrdil príjem ekvivalentnej správy od nás), nasleduje posielanie Update správy so sekvenčným číslom 2 a so smerovacou informáciou (vyjadrené príznakom Route). Potvrdzovacie číslo je nastavené rovnako, ako v prípade predchádzajúcej Update správy s Init príznakom (0 ak netreba potvrdiť prijatie správy, hodnota sekvenčného čísla z tabuľky susedov ak treba potvrdiť).

Posledným krokom, ktorý sa vykonáva iba v prípade, že je potrebné ešte potvrdiť prijatie nejakej potvrdzovanej správy od suseda, je poslanie Hello správy s nastaveným potvrdzovacím číslom.

Prechod *Enkapsuluj L3* slúži na enkapsuláciu smerovacej správy do L3 paketu. Ako vstup do prechodu slúži aktuálna správa na poslanie z miesta *Na poslanie* a záznam o susedovi z miesta *Tabulka susedov*. Výstupy po vykonaní tohto prechodu sú nasledovné:

- IPv4 paket vytvorený enkapsuláciou smerovacej správy s nastavenou zdrojovou a cieľovou adresou je uložený do miesta *Pridane L3*
- do miesta *Na poslanie* sa vygeneruje nový paket volaním generujúcej funkcie opísanej vyššie
- ak poslaná správa mala nastavené potvrdzovacie číslo, vynulujeme sekvenčné číslo v tabuľke susedov (potvrdili sme prijatie správy susedovi)



Obrázok 13: Trieda 3 – Enkapsulácia IPv4 paketu a rámca

IPv4 paket, ktorý je reprezentovaný farbou *PAKET* (miesto *Pridane L3*, znázornené na obrázku Obrázok 13) je definovaný v CPN ML nasledovne:

```
colset PAKET = record srcadd:INT * dstadd:INT * payload:SmerPAKET;
```

kde *srcadd* je zdrojová IPv4 adresa, *dstadd* je cieľová IPv4 adresa a *payload* predstavuje telo paketu, ktoré obsahuje posielanú smerovaciu správu.

Následná enkapsulácia IPv4 paketu do rámca je reprezentovaná prechodom *Enkapsuluj L2*, ktorý má ako vstup IPv4 paket z miesta *Pridane L3* a výstupom je rámec, ktorý je uložený do miesta *Pridany L2*. Samotný rámec je pomocou CPN ML jazyka definovaný nasledovne:

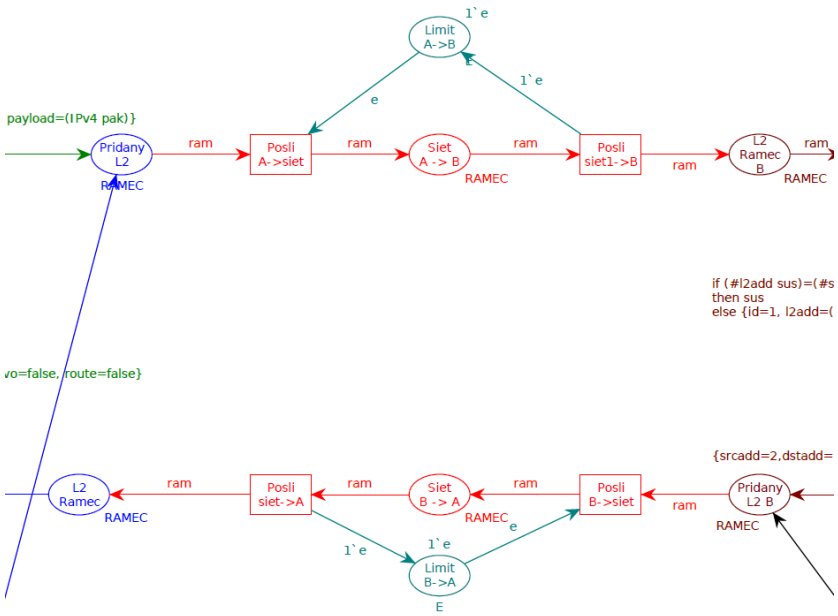
```
colset RAMEC = record srcadd:INT * dstadd:INT * payload:L3PAKET;
```

kde srcaddr je reprezentácia zdrojovej MAC adresy v rámci, dstadd je cieľová MAC adresa a payload reprezentuje telo rámca, zastúpené farbou *L3PAKET*. *L3PAKET* je definovaný nasledovne:

```
colset L3PAKET = union IPv4:PAKET + IPv6:PAKET6;
```

pričom jeho definícia nám umožní uložiť do tela rámca IPv4 ako aj IPv6 paket. Napríklad na hrane vedúcej z prechodu *Enkapsuluj L2* do miesta *Pridany L2* je rámec vytváraný výrazom `{srcadd=1,dstadd=2, payload=(IPv4 pak)}`, čo indikuje prítomnosť IPv4 paketu v rámci. Z definície paketu ako aj rámca je zrejmé, že adresy (IP aj MAC) sú reprezentované farbou INT, čo je celočíselná hodnota. Pre sprehľadnenie simulácie a modelu boli adresy IP aj MAC nastavené nasledovne: ľavý smerovač v modeli hodnota 1, pravý smerovač hodnota 2. Následne je rámec poslaný cez sieť ku druhému smerovaču.

6.1.2 Sieť medzi smerovačmi



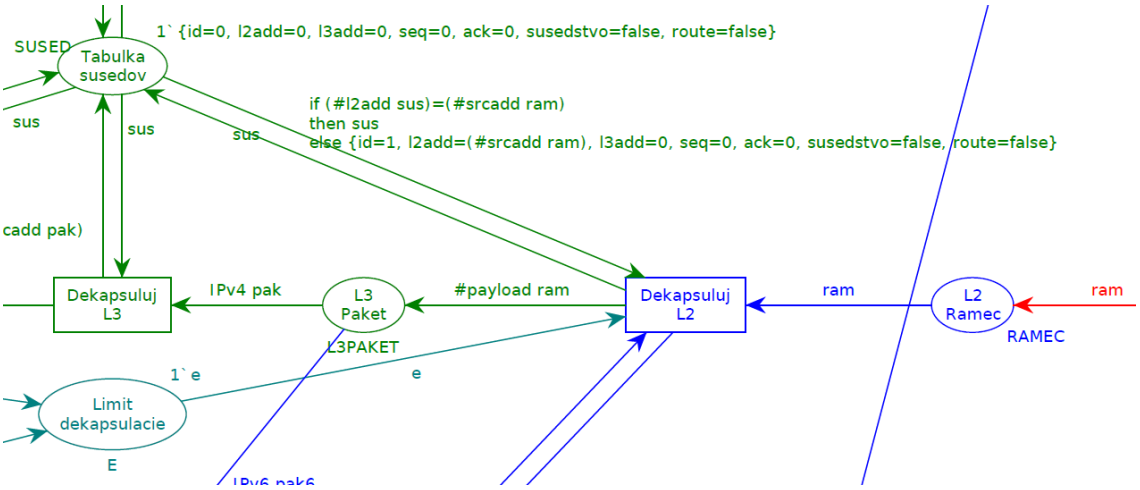
Obrázok 14: Trieda 3 – Reprezentácia siete

Časť modelu, ktorá reprezentuje sieť prenášajúcu rámce medzi smerovačmi je zobrazená na obrázku (Obrázok 14). Každý smer je reprezentovaný dvomi prechodmi a jedným miestom. Smer od smerovača A ku smerovaču B je namodelovaný nasledovne: Prechod *Posli A->siet* má ako vstup rámec od smerovača A z miesta *Pridany L2* a výstupom je rámec, ktorý je následne uložený do miesta *Siet A->B*. Potom je rámec zo siete preposlaný smerovaču B do miesta *L2 Ramec B* pomocou prechodu *Posli siet1->B*.

Ekvivalentne je namodelovaný smer od smerovača B ku smerovaču A, ktorý pozostáva z prechodov *Posli siet->B* a *Posli B->siet* a miesta *Siet B->A*.

6.1.3 Prijímacia časť

Prijímacia časť vytvoreného modelu zodpovedá za prijatie rámca, jeho dekapsuláciu a spracovanie.

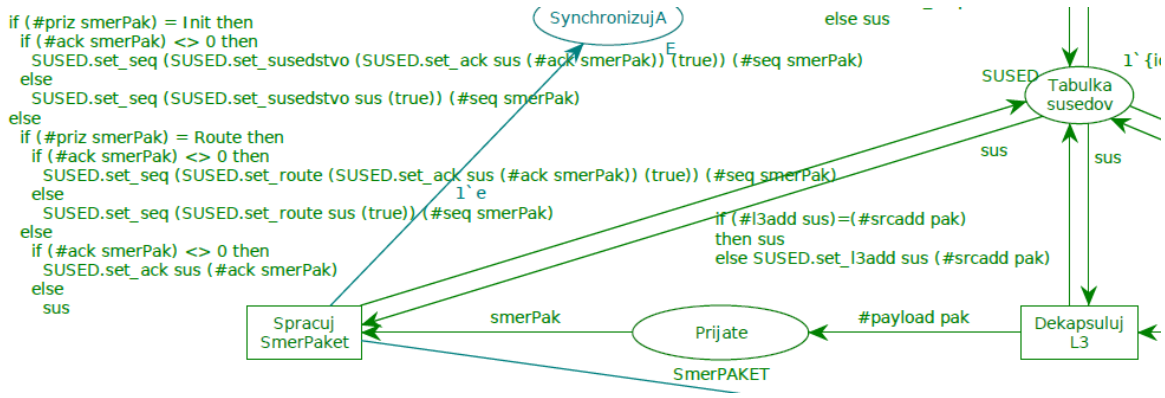


Obrázok 15: Trieda 3 – Prijatie a dekapsulácia rámca

Časť modelu zodpovedná za dekapsuláciu rámca je znázornená na obrázku (Obrázok 15). Najskôr sa prijatý rámec zo siete uloží do miesta *L2 Ramec*. Následne je spracovaný prechodom *Dekapsuluj L2*, ktorý z rámca vyberie IP paket a uloží ho do miesta *L3 Paket*. Ak sa jedná o IPv4 paket, tento je následne spracovávaný prechodom *Dekapsuluj L3*. Pri aktivácii prechodu *Dekapsuluj L2* je upravená aj *Tabulka susedov* nasledujúcou funkciou:

```
if (#l2add sus)=(#srcadd ram)
then sus
else {id=1, l2add=(#srcadd ram), l3add=0, seq=0, ack=0, susedstvo=false,
route=false}
```

Úlohou tejto funkcie je, v prípade, že sa v mieste *Tabulka susedov* ešte nenachádza záznam o susedovi, tento záznam vytvoriť s identifikátorom suseda a s MAC adresou prijatou od druhého smerovača.



Obrázok 16: Trieda 3 – Dekapsulácia IPv4 paketu a spracovanie smerovacej správy

Prechod *Dekapsuluj L3* spracúva IPv4 paket tak, že vyberie z neho smerovaciu správu a uloží ju do miesta *Prijate*. Ak záznam o susedovi ešte nemá nastavenú IPv4 adresu,

bude po aktivovaní tohto prechodu nastavený na hodnotu zdrojovej adresy z paketu výrazom:

```
if (#l3add sus)=(#srcadd pak)
then sus
else SUSED.set_l3add sus (#srcadd pak)
```

Posledným prechodom spracúvajúcim prijatú správu smerovacieho protokolu je prechod *Spracuj SmerPaket*. Na základe informácií v prijatej správe nastaví požadované položky záznamu v mieste *Tabulka susedov* nasledujúcou funkciou:

```
if (#priz smerPak) = Init then
  if (#ack smerPak) <> 0 then
    SUSED.set_seq (SUSED.set_susedstvo (SUSED.set_ack sus (#ack smerPak))
       (true)) (#seq smerPak)
  else
    SUSED.set_seq (SUSED.set_susedstvo sus (true)) (#seq smerPak)
else
  if (#priz smerPak) = Route then
    if (#ack smerPak) <> 0 then
      SUSED.set_seq (SUSED.set_route (SUSED.set_ack sus (#ack smerPak))
       (true)) (#seq smerPak)
    else
      SUSED.set_seq (SUSED.set_route sus (true)) (#seq smerPak)
  else
    if (#ack smerPak) <> 0 then
      SUSED.set_ack sus (#ack smerPak)
    else
      sus
```

V prvej vetve funkcie rozlišujeme, či bola prijatá správa s Init príznakom, čomu vyhovuje len Update správa posielaná pri vytváraní susedstva. V tomto prípade vykonáme nasledujúce zmeny v zázname tabuľky susedov:

- sekvenčné číslo v tabuľke nastavíme podľa sekvenčného čísla prijatej správy
- položku susedstvo nastavíme na hodnotu true
- potvrdzovacie číslo v tabuľke nastavíme v prípade, že potvrdzovacie číslo v prijatej správe bolo nenulové

V druhej vetve funkcie zisťujeme, či sa jedná o správu s nastaveným príznakom Route. Tejto podmienke vyhovuje len Update paket, ktorý reprezentuje prijaté smerovacie záznamy od suseda. V tomto prípade vykonáme nasledujúce zmeny v tabuľke susedov:

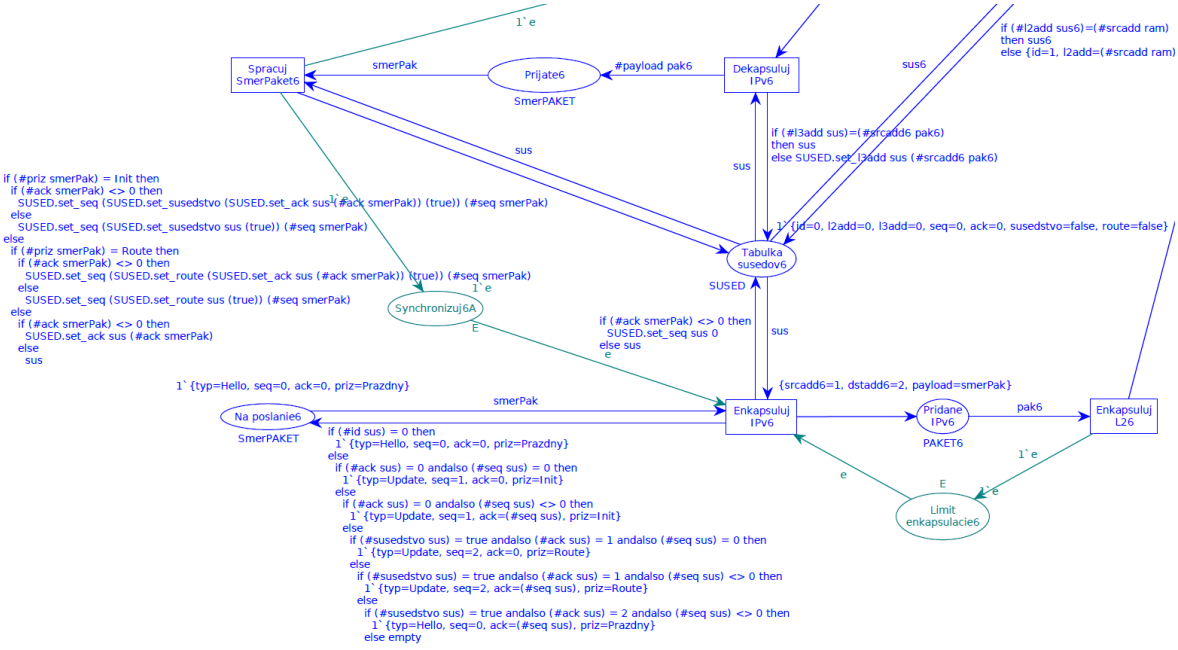
- sekvenčné číslo v tabuľke nastavíme podľa sekvenčného čísla prijatej správy
- položku route nastavíme na hodnotu true
- potvrdzovacie číslo v tabuľke nastavíme v prípade, že potvrdzovacie číslo v prijatej správe bolo nenulové

Ako posledné rozlišujeme prijatie Hello správy. V prípade, že potvrdzovacie číslo v prijatej správe bolo nenulové, nastavíme potvrdzovacie číslo v tabuľke susedov na hodnotu z prijatej správy. Inak tabuľku susedov nemeníme.

Keďže prijatú smerovaciu správu už ďalej spracovávať nepotrebujeme (všetky potrebné informácie z nej boli uložené v tabuľke susedov), správa v modeli zaniká.

6.1.4 Spracovanie IPv6

V predchádzajúcich podkapitolách boli popísané jednotlivé časti modelu, ktoré reprezentujú funkcionalitu spoločnú pre oba sieťové protokoly (spracovanie rámca a jeho prenos po sieti k ďalšiemu smerovaču) ako aj funkcionalitu špecifickú pre sieťový protokol IPv4. Časť modelu, ktorá je špecifická pre podporu IPv6 sieťového protokolu v smerovacom protokole tretej triedy je znázornená na obrázku (Obrázok 17).



Obrázok 17: Trieda 3 – Podpora IPv6

Ako je z obrázka zrejmé (Obrázok 17), časť siete reprezentujúca podporu IPv6 je veľmi podobná IPv4 časti. Zameriame sa preto hlavne na popis odlišných prvkov a spôsobu napojenia na všeobecnú časť modelu (spracovanie rámcov a poslanie cez sieť). Miesta, ktoré majú rovnakú funkciu ako v prípade IPv4 podmodelu, majú v IPv6 časti za názvom číslicu

6, pretože v jednom modeli nie je možné mať viaceré prvky s rovnakým názvom (model potom nie je možné verifikovať).

Odosielacia časť opäť pozostáva z miesta *Na poslanie6*, prechodu *Enkapsuluj IPv6*, ktorý má funkciu rovnakú ako prechod *Enkapsuluj L3* v prípade IPv4, akurát funkcia na výstupnej hrane je definovaná nasledovne:

```
{srcadd6=1, dstadd6=2, payload=smerPak}
```

Ako je z definície zrejmé, jednotlivé premenné reprezentujúce adresy majú za názvom 6, pretože v tomto prípade funkcia vytvára farbu s názvom *PAKET6* a nasledujúcou definíciou:

```
colset PAKET6 = record srcadd6:INT * dstadd6:INT * payload:SmerPAKET;
```

Na enkapsuláciu paketu sa opäť využívajú informácie z miesta *Tabulka susedov6*, rovnakým spôsobom ako v prípade IPv4 časti. Následne sa IPv6 paket spracúva v prechode *Enkapsuluj L26*, kde je vytvorený rámec, v tomto prípade ale nasledujúcim výrazom:

```
{srcadd=1,dstadd=0, payload=(IPv6 pak6)}
```

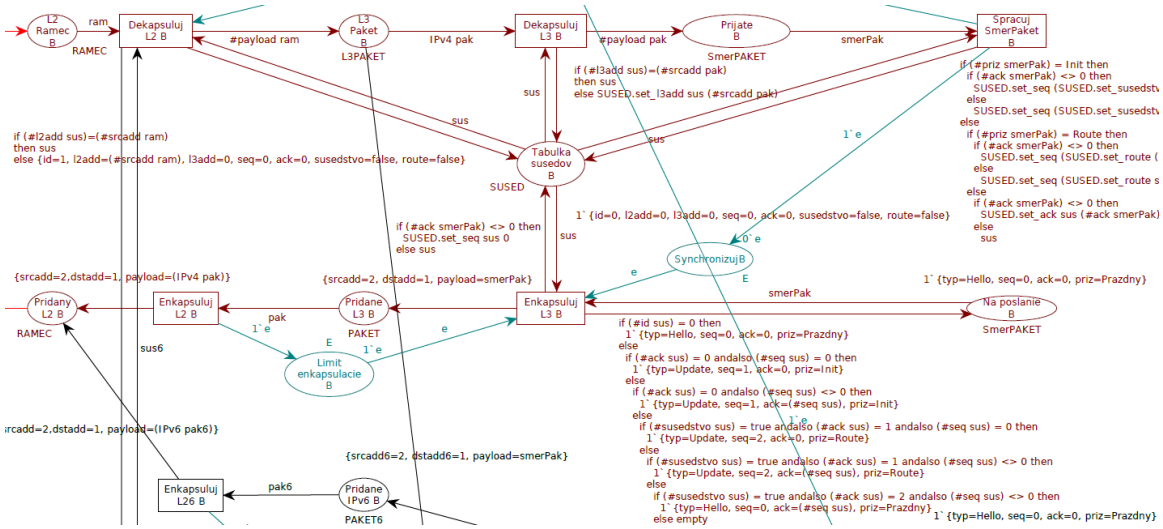
Premenná *payload* v tomto prípade identifikuje v rámci prítomnosť IPv6 paketu. Paket je uložený v mieste *Pridany L2*, rovnako ako v prípade IPv4.

Prijímacia časť IPv6 opäť pozostáva z prechodu *Dekapsuluj IPv6*, kde sa IPv6 paket dostane z miesta *L3 Paket*. Na základe údajov v pakete je podobne ako v prípade IPv4 upravená IPv6 adresa suseda v mieste *Tabulka susedov6*. Z paketu sa vyberie smerovacia správa a uloží sa do miesta *Prijate6*. Výsledná smerovacia správa sa spracúva prechodom Spracuj *SmerPaket6*, úplne rovnako, ako v prípade IPv4 časti modelu.

Vďaka vytvoreniu separátnej časti modelu pre spracovanie IPv6, popísanej v tejto podkapitole, je možná reprezentácia vytvárania dvoch samostatných susedských relácii, jedna pre IPv4 protokol a druhá pre IPv6 protokol.

6.1.5 Reprezentácia smerovača B

Grafické znázornenie spoločných a IPv4 častí smerovača B je zobrazené na obrázku (Obrázok 18).



Obrázok 18: Trieda 3 – Smerovač B spoločná časť a IPv4

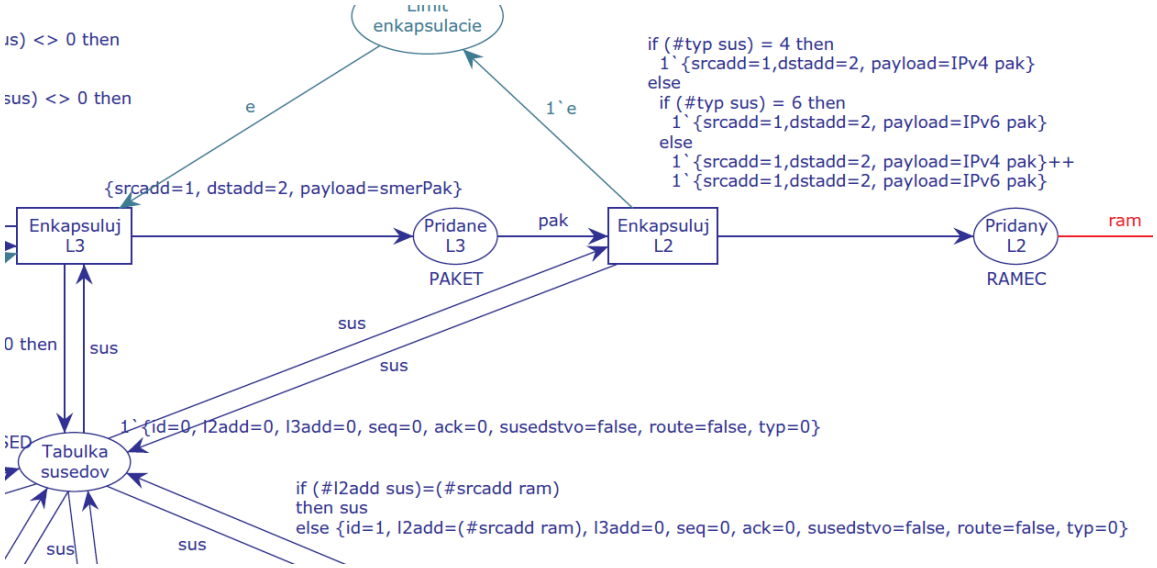
Ako si môžeme na obrázku všimnúť, smerovač B je kópiou časti modelu smerovača A, so zmenenými názvami miest a prechodov (pridané písmeno B na konci). Taktiež boli zmenené L2 a L3 adresy v paketoch a rámcoch a tabuľkách susedov, keďže smerovač B má adresu nastavenú na hodnotu 2. IPv6 časť je upravená ekvivalentne a viditeľná v celej grafickej reprezentácii suseda B zobrazenej v prílohe G.

6.2 Štvrtá trieda – transport L3, jedno susedstvo

Model štvrtej triedy riešenia protokolovej integrovanosti sme odvodili od úplného modelu tretej triedy. Celá grafická reprezentácia vytvoreného modelu znázorňujúceho štvrtú triedu je zobrazená v prílohách H (časť reprezentujúca sieť a smerovač A) a I (časť reprezentujúca sieť a smerovač B). V prvom kroku sme z modelu odstránili časti špecifické pre IPv6, keďže model štvrtej triedy vytvára iba jedno spoločné susedstvo. Následne sme sa zaoberali modifikáciou odosielacej časti.

6.2.1 Odosielacia časť

Odosielacia časť opäť pozostáva z miest Na poslanie, Tabulka susedov, Pridane L3, Pridany L2 a prechodov Enkapsuluj L3, Enkapsuluj L2, ktoré ale boli modifikované ako je znázornené na obrázku (Obrázok 19).



Obrázok 19: Trieda 4 – Zmeny v odosielacej časti

Prvým problémom, bol fakt, že v modeli triedy štyri vytvárame iba jedno susedstvo a prenos správ prebieha na sieťovej vrstve, takže smerovacie správy môžu byť enkapsulované do IPv4 aj IPv6 paketu. Smerovače ale musia komunikovať rovnakou verziou IP protokolu, keďže navzájom nie sú „kompatibilné“. Ako najvšeobecnejšie riešenie nám po rôznych pokusoch vyšiel nasledujúci algoritmus:

Prvá smerovacia správa sa pošle dvakrát, pričom jedna bude zabalená do IPv4 paketu a druhá do IPv6 paketu a poslaná cez sieť (predpokladáme, že smerovač má zapnutú podporu oboch sieťových protokolov). Druhý smerovač, na základe toho, ktorý paket prijme skôr, bude s týmto susedom komunikovať iba pomocou sieťového protokolu, ktorého paket prijal ako prvý. Druhý smerovač bude postupovať ekvivalentne. Optimalizáciou tohto algoritmu môže byť poslanie prvého z týchto dvoch paketov IPv6, aby sa minimalizovali situácie, kde každý zo smerovačov zašle ako prvý paket s rôznym sieťovým protokolom. Týmto algoritmom zaistíme, že smerovače vedia spolu komunikovať pomocou spoločného sieťového protokolu a nič nebráni nadviazaniu susedstva a následnej výmene smerovacích informácii (oboch podporovaných sieťových protokolov).

Pre namodelovanie navrhnutého algoritmu boli potrebné nasledujúce zmeny v odosielacej časti. Najskôr sme do tabuľky susedov pridali položku signalizujúcu, susedstvo s ktorým sieťovým protokolom sme nadviazali (hodnota 4 pre IPv4 a 6 pre IPv6 protokol). Výsledná upravená definícia položky tabuľky susedov je nasledovná:

```
colset SUSED = record id:INT * l2add:INT * l3add:INT * seq:INT * ack:INT *
susedstvo:BOOL * route:BOOL * typ:INT;
```

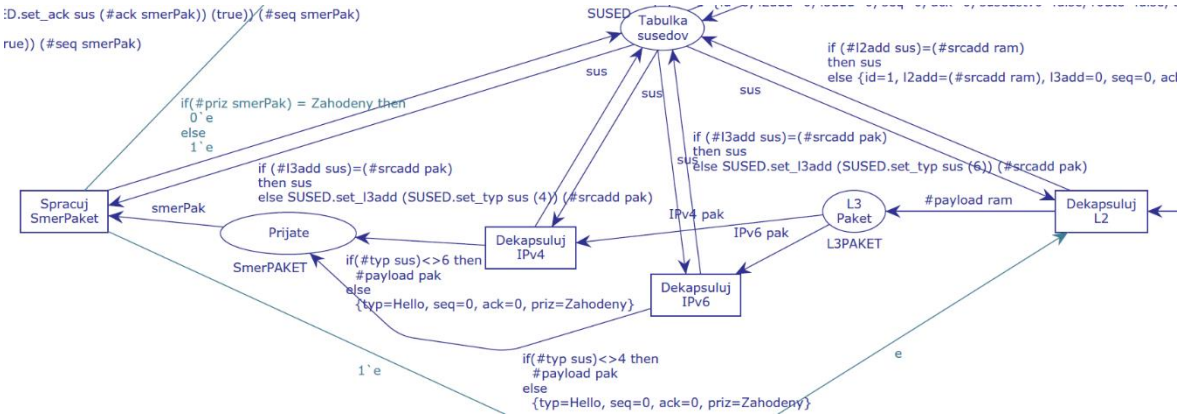
Následne sme upravili funkciu na hrane z prechodu *Enkapsuluj L2* do miesta *Pridany L2* nasledovne:

```
if (#typ sus) = 4 then
  1`{srcadd=1,dstadd=2, payload=IPv4 pak}
else
  if (#typ sus) = 6 then
    1`{srcadd=1,dstadd=2, payload=IPv6 pak}
  else
    1`{srcadd=1,dstadd=2, payload=IPv4 pak}++
    1`{srcadd=1,dstadd=2, payload=IPv6 pak}
```

V prípade, že v tabuľke susedov je typ susedstva nastavený na hodnotu 4 (prijali sme ako prvý paket s IPv4 protokolom) funkcia bude do rámcov baliť len IPv4 pakety. Ak je ale typ nastavený na hodnotu 6, do rámcov pôjdu len IPv6 pakety. Ak je typ nastavený na inú hodnotu ako 4 alebo 6 (ešte sme neprijali paket od susedného smerovača), vytvoria sa dva rámce, jeden s IPv4 a druhý s IPv6 paketom.

6.2.2 Prijímacia časť

Prijímaciu časť modelu štvrtej triedy riešenia protokolovej integrovanosti bolo potrebné modifikovať pre uplatnenie algoritmu popísaného v podkapitole 6.1.1. Modifikovaná časť je zobrazená na obrázku (Obrázok 20).



Obrázok 20: Trieda 4 – Zmeny v prijímacej časti

Prijatý rámec je dekapsulovaný prechodom *Dekapsuluj L2* a následne uložený do miesta *L3 Paket*.

Následne sú IPv4 pakety spracované prechodom *Dekapsuluj IPv4*. Ak sa jedná o prvý paket od suseda, do tabuľky susedov sa nastaví typ susedstva na hodnotu 4 a uloží sa IP adresa suseda. Ak sa nejedná o prvý paket od suseda a už bol prijatý IPv6 paket, tento

paket sa zahodí a vygeneruje sa prázdny smerovací paket s príznakom *Zahodeny*. Paket sa nakoniec uloží do miesta *Prijate* a neskôr je spracovávaný prechodom *Spracuj SmerPaket*.

V prípade prijatia IPv6 paketu ako prvého je situácia rovnaká, akurát je využitý prechod *Dekapsuluj IPv6*, ktorý nastavuje hodnoty špecifické pre IPv6 (hodnota typu v tabuľke susedov s číslom 6).

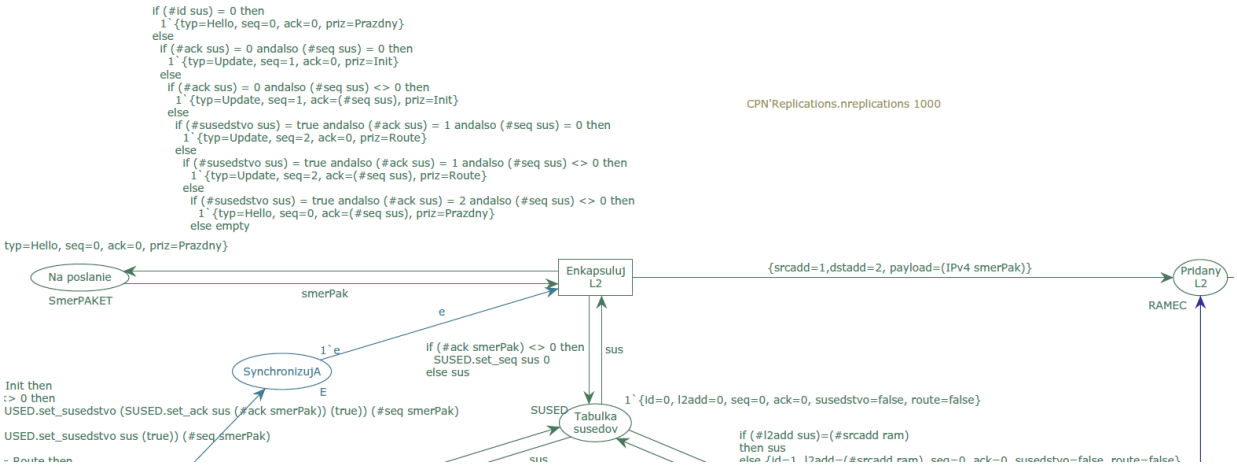
Časť modelu reprezentujúca smerovač B bola vytvorená rovnakým spôsobom, ako bolo predstavené v podkapitole 6.1.5. Bolo potrebné vytvoriť kópiu časti smerovača A spolu so zmenami adries.

6.3 Prvá trieda – transport L2, viaceré susedstvá

Model prvej triedy riešenia protokolovej integrovanosti sme rovnako, ako v prípade štvrtej triedy (kapitola 6.2) odvodili od úplného modelu tretej triedy (kapitola 6.1). Celá grafická reprezentácia vytvoreného modelu znázorňujúceho prvú triedu je zobrazená v prílohách B (časť reprezentujúca sieť a smerovač A) a C (časť reprezentujúca sieť a smerovač B). Keďže prvá trieda prenáša smerovacie správy priamo zabalené v rámcoch, museli sme odstrániť všetky prvky modelu, ktoré vkladali smerovacie správy do paketov a tie následne do rámcov. Najskôr sme sa zaoberali modifikáciou odosielacej časti.

6.3.1 Odosielacia časť

Grafické znázornenie modifikovanej odosielacej časti je zobrazené na obrázku (Obrázok 21).



Obrázok 21: Trieda 1 – Zmeny v odosielacej časti

Odosielacia časť pozostáva z miest *Na poslanie*, *Tabulka susedov*, *Pridany L2* a prechodu *Enkapsuluj L2*. Z položky tabuľky susedov bola odobratá premenná ukladajúca

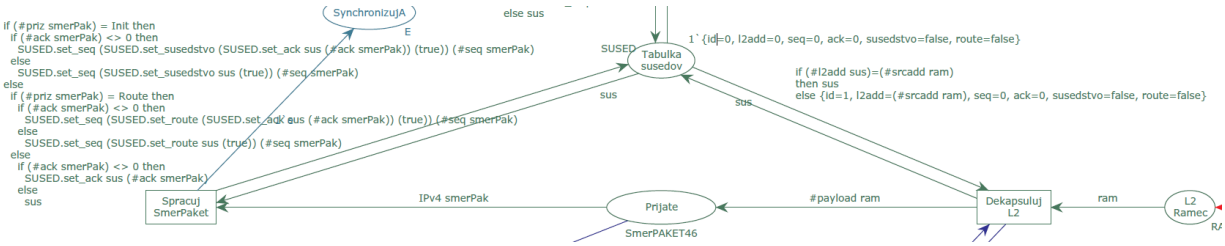
IP adresu suseda. Generovanie smerovacích správ zostalo nezmenené. Najdôležitejšia zmena bola vykonaná prepojením miesta *Na poslanie* priamo s prechodom *Enkapsuluj L2*, pričom výstup z prechodu je rámec, kde *payload* je smerovacia správa pre IPv4 susedstvo vytvorená nasledovnou funkciou:

```
{srcadd=1,dstadd=2, payload=(IPv4 smerPak)}
```

kde výraz *(IPV4 smerPak)* indikuje, že v rámci je nesená práve smerovacia správa IPv4 susedstva (nejedná sa ale o IPv4 paket na sieťovej vrstve).

6.3.2 Prijímacia časť

Grafická podoba modifikovanej prijímacej časti je na obrázku (Obrázok 22).



Obrázok 22: Trieda 1 – Zmeny v prijímacej časti

Prijímacia časť pozostáva z miest *L2 Ramec*, *Prijate*, *Tabulka susedov* a prechodov *Dekapsuluj L2*, *Spracuj SmerPaket*. Z rámca je prechodom *Dekapsuluj L2* vybratá smerovacia správa a uložená do miesta *Prijate*. Súčasne sa aktualizuje tabuľka susedov o informácie z rámca, ak je to potrebné. V mieste Prijate sa môžu uložiť správy pre IPv4 aj IPv6 susedstvo, keďže jeho farba je *SmerPAKET46* s nasledovnou definíciou:

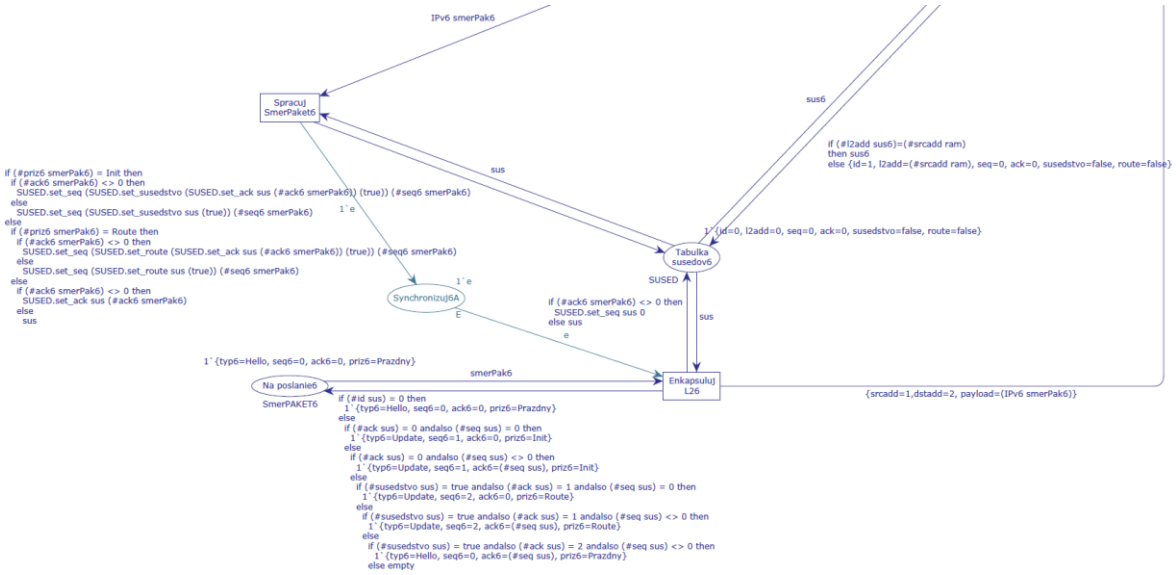
```
colset SmerPAKET46 = union IPv4:SmerPAKET + IPv6:SmerPAKET6;
```

Ak sa jednalo o smerovaciu správu IPv4 susedstva, správa sa následne spracuje prechodom *Spracuj SmerPaket*. Po spracovaní je správa zahodená.

6.3.3 Spracovanie IPv6 susedstva

Keďže prvá trieda riešenia protokolovej integrovanosti vytvára dve osobitné susedstvá pre oba sieťové protokoly, časť modelu popísanú v predchádzajúcich podkapitolách sme zduplikovali a upravili pre spracovanie IPv6 susedstva. Táto časť modelu je viditeľná na obrázku (Obrázok 23). Názvy miest a prechodov sme museli upraviť pridaním číslice 6 na koniec názvu.

Odosielacia časť po enkapsulovaní rámca prechodom *Enkapsuluj L26* uloží vytvorený rámec do miesta *Pridany L2* rovnako, ako IPv4 časť. IPv6 časť má vlastnú databázu susedov reprezentovanú miestom *Tabulka susedov6*.



Obrázok 23: Trieda 1 – Zmeny v časti pre IPv6 susedstvo

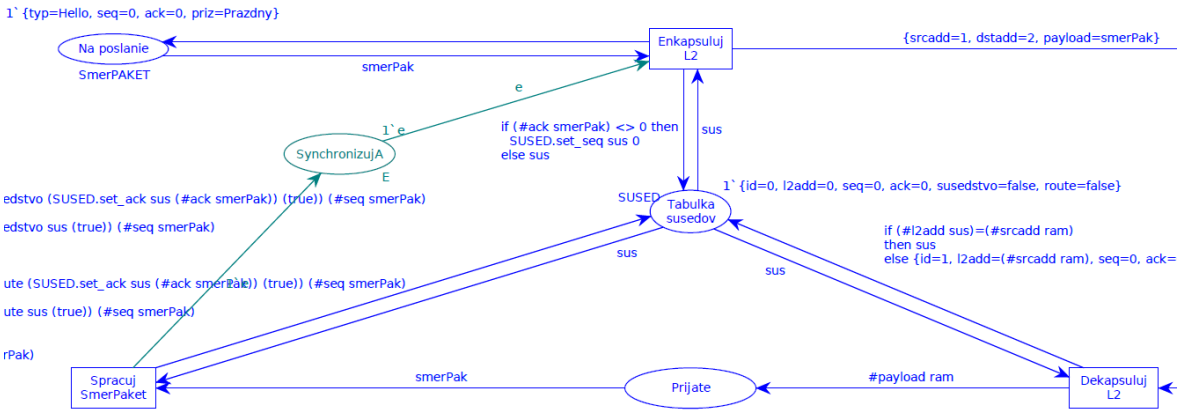
Prijímacia časť zdieľa miesta *L2 Ramec*, *Prijate* a prechod *Dekapsuluj L2* s IPv4 časťou. Samostatné sú len miesto *Tabulka susedov6* a prechod *Spracuj SmerPaket6*, do ktorého sa smerovacia správa IPv6 susedstva dostane z miesta *Prijate*. Definícia smerovacej správy pre IPv6 susedstvo je reprezentovaná farbou *SmerPAKET6* a vyzerá nasledovne:

```
colset SmerPAKET6 = record typ6:TYP * seq6:SEQ * ack6:SEQ * priz6:PRIZNAK;
```

Časť modelu reprezentujúca smerovač B bola vytvorená rovnakým spôsobom, ako bolo predstavené v podkapitole 6.1.5. Bolo potrebné vytvoriť kópiu časti smerovača A, a následne zmeniť adresy. K názvom prechodov a miest bolo pridané písmeno B, keďže v jednom modeli sa nesmú nachádzať prvky s rovnakým názvom.

6.4 Druhá trieda – transport L2, jedno susedstvo

Model druhej triedy riešenia protokolovej integrovanosti sme odvodili od úplného modelu prvej triedy (kapitola 6.3), keďže s ním zdieľa najviac podobností. Celá grafická reprezentácia vytvoreného modelu znázorňujúceho druhú triedu je zobrazená v prílohách D (časť reprezentujúca sieť a smerovač A) a E (časť reprezentujúca sieť a smerovač B). Keďže druhá trieda vytvára iba jedno spoločné susedstvo, odstránili sme všetky prvky zabezpečujúce IPv6 susedstvo z kópie modelu prvej triedy. Časť výsledného modelu reprezentujúca smerovač A je znázornená na obrázku (Obrázok 24).



Obrázok 24: Trieda 2 – Reprezentácia smerovača A

Ako vidíme na obrázku (Obrázok 24), výsledný model smerovača A je najjednoduchší spomedzi modelov všetkých tried riešenia protokolovej integrovanosti. Odosielacia časť pozostáva len z miest *Na poslanie*, *Tabulka susedov*, *Pridany L2* a prechodu *Enkapsuluj L2*. Prijímacia časť obsahuje miesta *L2 Ramec*, *Tabulka susedov*, Prijate a prechody *Dekapsuluj L2* a *Spracuj SmerPaket*.

Reprezentácia smerovača B bola opäť vytvorená ako kópia časti modelu smerovača A, pričom na koniec názvu všetkých prvkov bolo pridané písmeno B a boli zmenené zdrojové a cieľové L2 adresy.

7 Verifikácia vytvorených modelov

Po vytvorení prvého modelu (tretej triedy riešenia protokolovej integrovanosti, popísaného v kapitole 6.1) bolo nutné overiť, či vytvorený model spĺňa funkčné požiadavky, medzi ktoré patrí vytvorenie susedstva medzi smerovačmi, prítomnosť očakávaných hodnôt v tabuľke susedov, generovanie správnych správ na základe hodnôt v tabuľke susedov a iných.

7.1.1 Problém výpočtu stavového priestoru a automatickej simulácie

Ako prvý krok overenia návrhu, na základe vykonanej analýzy farbených Petriho sietí popísanej v kapitole 3.3 a nástroja CPN Tools popísaného v kapitole 3.4, sme sa pokúsili vykonať verifikáciu modelu pomocou vypočítania stavového priestoru. Bohužiaľ, výpočet bol ukončený po 5 minútach (štandardná hodnota ako dlho sa má počítať stavový priestor), pričom nás nástroj informoval, že bol vypočítaný iba čiastočný stavový priestor. Preto sme zvýšili dĺžku výpočtu na 30 minút, ale opäť sme dostali informáciu o iba čiastočnom výpočte stavového priestoru, pričom ale počet uzlov a hrán stavového priestoru násobne narástol. Ani po zvýšení výpočtového času na 1 deň sme nedostali informáciu o plnom stavovom priestore. Keďže sme ani po ďalšom štúdiu literatúry a predlžovaní času výpočtu nedokázali zistiť problém iba čiastočného výpočtu stavového priestoru, pokúsili sme sa odskúšať model iným spôsobom.

Podobným spôsobom, ako sme overovali funkčnosť častí modelu už počas vytvárania, sme sa rozhodli overiť celý vytvorený model. Vykonali sme interaktívnu simuláciu vytvoreného modelu tak, že sme manuálne vyberali z aktívnych prechodov tie, ktoré sa majú vykonať v nasledovnom poradí: od IPv4 časti smerovača A -> enkapsulácia smerovacej správy do paketu -> enkapsulácia paketu do rámca -> poslanie cez sieť -> prijatie rámca susedom B -> dekapsulácia paketu z rámca -> dekapsulácia smerovacej správy -> spracovanie správy -> poslanie úvodnej správy susedom B a tak ďalej až pokým si smerovače nevymenili všetky požadované správy, a teda nenadviazali susedstvo a vymenili si smerovacie informácie. Priebeh bol teda podobný ako bolo znázornené na obrázku (Obrázok 8), reprezentujúcom vytvorenie susedstva smerovacím protokolom EIGRP. Proces dopadol podľa očakávania, preto sme ho opakovali pre IPv6 susedstvo s opätovným úspechom. Interaktívna simulácia IPv4 aj IPv6 si vyžiadala okolo 250 simulačných krokov, čo bolo časovo náročné (bolo treba 250 krát vybrať požadovaný prechod). Keďže

interaktívna simulácia dopadla úspešne, rozhodli sme sa pokračovať v preskúmavaní možností simulácie.

Ako nasledujúci krok sme sa pokúsili vykonať automatickú simuláciu nastavením počtu vykonaných krokov na 300, keďže sme očakávali ukončenie simulácie po 250 krokoch ako v prípade interaktívnej simulácie. Avšak aj napriek sľubným výsledkom interaktívnej simulácie, automatická simulácia neskončila ani po nastavení krokov na počet v 10000, podobne ako v prípade výpočtu stavového priestoru.

Tento fakt nás viedol k hlbšiemu preskúmaniu vzorových modelov v literatúre [12], [26] a dokumentácie na oficiálnej stránke nástroja CPN Tools [32], ktorými sme sa zaoberali už počas predchádzajúcej analýzy, keďže vedecké články, ktoré sme využívali počas analýzy nepopisovali do detailov spôsob modelovania, verifikovania a simulácie farbených Petriho sietí. Dodatočné štúdium týchto zdrojov a výpočet stavového priestoru a simulácia vzorových modelov farbených Petriho sietí pre nástroj CPN Tools nás priviedlo k riešeniu popisovaného problému.

7.1.2 Identifikácia problému výpočtu plného stavového priestoru a automatickej simulácie

Problém bol spôsobený situáciou, že veľký počet prechodov v modeli bol pri určitých krokoch vykonávania modelu aktívny. Ako príklad môžeme uviesť situáciu, že len v prvom kroku vykonávania navrhnutého modelu tretej triedy riešenia protokolovej integrovanosti sa môže vykonať jeden z nasledujúcich štyroch prechodov:

- Enkapsuluj L3
- Enkapsuluj IPv6
- Enkapsuluj L3 B
- Enkapsuluj IPv6 B

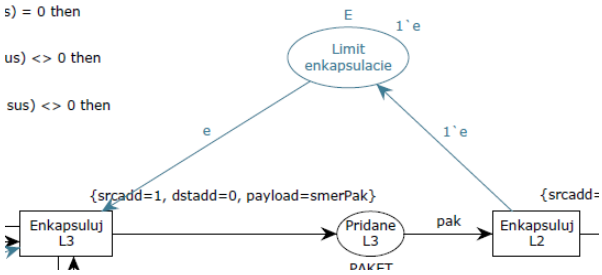
Preto je zrejmé, že už len kombináciou aktívnych prechodov narastá stavový priestor o veľmi veľký počet uzlov a hrán.

Druhý z identifikovaných problémov spôsobených neriadeným počtom aktivovaných prechodov je situácia, keď by sa napríklad vykonávali stále iba prechody posielajúce úvodnú správu IPv4 susedstva od suseda A k smerovaču B. Keďže smerovač B by nikdy neodpovedal smerovaču A, nezačalo by sa vytvárať susedstvo a teda v modeli by nastalo uviaznutie.

Presne tieto popísané problémy spôsobili výpočet iba čiastočného stavového priestoru a „nekončiacu“ automatickú simuláciu. Vytvorený model teda nebol deterministický a bolo potrebné obmedziť počet aktívnych prechodov a synchronizovať oba susedné smerovače.

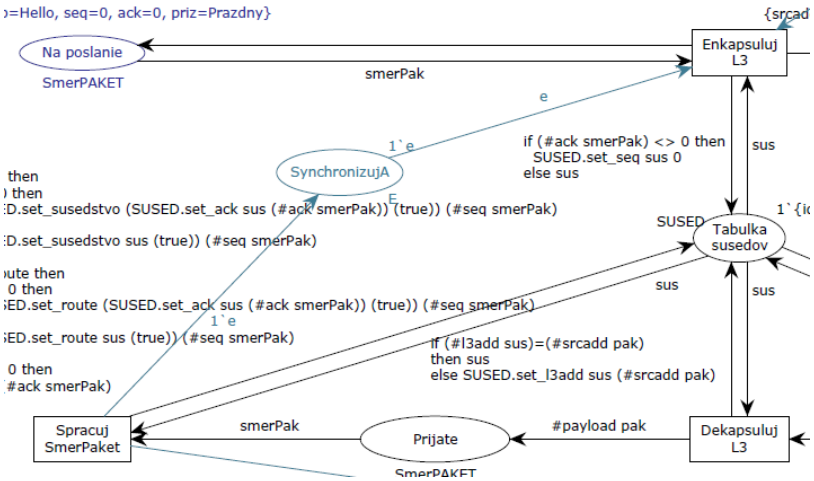
7.1.3 Riešenie problému výpočtu plného stavového priestoru a automatickej simulácie

Na základe vzorových modelov v literatúre [12], [26] a dokumentácie na oficiálnej stránke nástroja CPN Tools [32], sme vytvorili viaceré limitujúce a synchronizačné miesta, ktoré obmedzujú aktívne prechody v modeli.



Obrázok 25: Limitujúce miesto v modeli

Na obrázku (Obrázok 25) je znázornené limitujúce miesto s názvom *Limit enkapsulacie* (zvýraznené tyrkysovou farbou), ktoré zabezpečuje, že v jednom kroku môže byť prechodmi *Enkapsuluj L3* a *Enkapsuluj L2* spracovávaná iba jedna správa. To znamená, že prechod *Enkapsuluj L3* bude aktívny iba v prípade, že nie je aktívny prechod *Enkapsuluj L2*. Podobným spôsobom môžu byť aplikované limity aj na viaceré prechody (nie len dva). My sme aplikovali limitujúce miesto aj napríklad medzi prechodmi *Dekapsuluj L2* a *Spracuj SmerPaket*, čo znamená, že prechod *Dekapsuluj L2* bude môcť byť opäť aktívny až potom, ako sa vykoná prechod *Dekapsuluj L3* a *Spracuj SmerPaket*.



Obrázok 26: Synchronizačné miesto v modeli

Synchronizačné miesto s názvom *SynchronizujA*, zvýraznené tyrkysovou farbou je rovnako, ako limitujúce miesta, znázornené na obrázku (Obrázok 26). Toto miesto slúži na odstránenie problému uviaznutia modelu spôsobeného neustálym posielaním správ iba jedným smerovačom v modeli, ktorý ale nikdy nedostane odpoveď. Synchronizačné miesta boli vytvorené jedno pre každé susedstvo v reprezentácii každého zo smerovačov. Podobnú funkciu ako synchronizačné miesta v modeli plnia v reálnych implementáciách smerovacích protokolov časovače.

Po aplikovaní popísaných opráv do všetkých štyroch modelov tried riešenia protokolovej integrovanosti sa nám úspešne podarilo verifikovať všetky vytvorené modely.

7.1.4 Výsledky verifikácie modelov

Výsledkom výpočtu stavového priestoru každého modelu pomocou nástroja CPN Tools je textový súbor s informáciami a štatistikami. Na základe analýzy týchto výstupov sme boli schopní zistiť, či nenastalo uviaznutie v modeli a štatistické informácie ako napríklad, koľko značiek bolo maximálne prítomných v každom mieste.

Najdôležitejšia časť výstupu výpočtu stavového priestoru prvej triedy je nasledovná:

```
Statistics
------------------------------------------------------------------------

  State Space
     Nodes:  21787
     Arcs:   42190
     Secs:   6
     Status: Full


 Boundedness Properties
```

```
------------------------------------------------------------------------
  Best Integer Bounds
                             Upper      Lower
     Top'L2_Ramec 1          2          0
     Top'L2_Ramec_B 1        2          0
     Top'Na_poslanie 1       1          0
     Top'Na_poslanie6 1      1          0
     Top'Na_poslanie6_B 1    1          0
     Top'Na_poslanie_B 1     1          0
     Top'Pridany_L2 1        2          0
     Top'Pridany_L2_B 1      2          0
     Top'Prijate 1           2          0
     Top'Prijate_B 1         2          0
     Top'Siet_A 1            1          0
     Top'Siet_B 1            1          0
     Top'Tabulka_susedov 1   1          1
     Top'Tabulka_susedov6 1  1          1
     Top'Tabulka_susedov6_B1 1          1
     Top'Tabulka_susedov_B 1 1          1
Liveness Properties
------------------------------------------------------------------------
  Dead Markings [21765,21772,21787]
```

Odstavec State space obsahuje informácie o vypočítanom stavovom priestore. Počet uzlov vyjadruje položka Nodes, položka Arcs počet hrán a pre nás najdôležitejšia položka Status indikuje, či bol vypočítaný plný stavový priestor (teda simulácia bola ukončená po vybudovaní susedstva a výmene smerovacej informácie).

Ako si môžeme všimnúť z výpisu štatistiky, stavový priestor obsahuje desaťtisíce položiek aj napriek domu, že model pozostáva iba z 22 miest a 14 prechodov. Dôvodom je, že v modeli prvej triedy sú vytvárané obe susedstvá súčasne, čo zvyšuje počty možných kombinácií aktívnych prechodov.

Medzi ďalšie zaujímavé položky patria údaje v odstavci Best Integer Bounds, kde pre konkrétne miesta vidíme, koľko značiek maximálne a minimálne sa v nich súčasne vyskytovalo.

Ako poslednú zaujímavú časť výpisu výpočtu stavového priestoru uvedieme položku Dead Markings, ktorá označuje uzly stavového priestoru, v ktorých simulácia končí. V prípade modelu prvej triedy sú uzly 3, čo je opäť spôsobené dvomi susedstvami vytvárajúcimi sa súčasne.

Najdôležitejšia časť výstupu výpočtu stavového priestoru druhej triedy vyzerá nasledovne:

```
Statistics
------------------------------------------------------------------------
  State Space
     Nodes:  91
     Arcs:   90
     Secs:   0
     Status: Full
Liveness Properties
------------------------------------------------------------------------

  Dead Markings [91]
```

Keďže model druhej triedy vytvára iba jedno susedstvo a je jednoduchší než ostatné modely, jeho stavový priestor je menší, a taktiež obsahuje iba jeden uzol, v ktorom môže simulácia skončiť.

Stavový priestor tretej triedy je, čo sa týka vlastností, podobný ako model prvej triedy, keďže tiež vytvára dve susedstvá. Hlavný rozdiel je iba vo väčšom počte uzlov a hrán stavového priestoru.

Výpočet stavového priestoru štvrtej triedy prebehol rovnako úspešne ako u ostatných modelov. Aj napriek tomu, že vytvára jedno susedstvo, stavový priestor obsahuje viac uzlov a hrán ako v prípade modelu prvej triedy. Táto situácia je spôsobená riešením problému, ktorý sieťový protokol sa použije na výmenu smerovacích správ.

Plné textové výstupy výpočtu stavového priestoru modelu každej triedy sú uvedené v prílohe A.

8 Porovnanie jednotlivých modelov

Aby bolo možné vytvoriť metodiku pre dizajn integrovaných smerovacích protokolov, museli sme jednotlivé modely tried riešenia protokolovej integrovanosti rigorózne porovnať. Jednou z najväčších výziev bolo práve výber vhodného kritéria, na základe ktorého bude možné výsledky porovnania využiť na vytvorenie metodiky pre dizajn integrovaných protokolov, ktorá bude nezávislá od konkrétnej budúcej implementácie, preto musí byť vybraté kritérium dostatočne všeobecné. Preto sme vykonali analýzu dostupnej literatúry, ktorá sa zaoberá vytváraním modelov farbených Petriho sietí [12], [26], [32] ako aj rôznych vedeckých publikácii, zaoberajúcich sa využitím farbených Petriho sietí pre zlepšenie sieťových protokolov [10], [15], [38], [56]–[60], [16], [23]–[28], [34].

Ako prvé kritérium, nad ktorým sme uvažovali na základe jeho použitia pri simulácii rôznych modelov protokolov, je dĺžka nadviazania susedstva spolu s časom potrebným na následnú výmenu smerovacích informácií. Použitie časového kritéria predpokladá, že každý prechod v modeli má pridelenú svoju dĺžku trvania. Problém ale nastáva pri určovaní týchto časových intervalov, keďže dĺžka trvania je implementačne závislým parametrom. Ako príklad uvedieme prechod *Enkapsuluj L3*, ktorý v modeloch tretej a štvrtej triedy slúži na zabalenie smerovacej správy do IP paketu. Táto operácia môže byť v reálnej implementácii smerovača vykonávaná softvérovo, čo môže byť príklad lacných smerovačov, ktoré nie sú optimalizované na prevádzku vo vysokorýchlostných sieťach ale na cenu, pričom pri softvérovej implementácii môže trvať v rádoch mikrosekúnd. Naopak, táto operácia môže byť realizovaná hardvérovým prvkom, čo je prípad súčasných vysokorýchlostných smerovačov, kde táto operácia môže trvať v rádoch nanosekúnd až pikosekúnd. Ako vidíme, kritérium času je veľmi vhodné na porovnanie napríklad dvoch implementácií rovnakého smerovacieho protokolu, ale nie je dostatočne všeobecné pre porovnanie generických tried riešenia protokolovej integrovanosti.

Ako ďalšie možné kritérium porovnania modelov sa intuitívne naskytá použitie počtu miest alebo prechodov. Problémom tohto kritéria, je ale fakt, ktorý sme zistili na základe analýzy literatúry, že niektoré miesta alebo prechody mohli byť v modeli použité z dôvodu obmedzení modelovacieho jazyka. Problém môžeme ilustrovať aj na základe našej skúsenosti z modelovania prijímacej časti modelu štvrtej triedy zobrazenej na obrázku (Obrázok 20). Na dekapsulovanie IP paketov bolo potrebné vytvoriť samostatné prechody *Dekapsuluj IPv4* a *Dekapsuluj IPv6*, pretože modelovací jazyk neumožňuje spracovanie

dvoch rôznych prvkov štruktúry *union* na jednej hrane. Keďže nie je možné súčasné „*pretypovanie*“ a porovnanie jednotlivých podpoložiek „*pretypovanej*“ farby.

Ako tretie uvažované kritérium bolo porovnanie na základe počtu simulačných krokov potrebných na to, aby sa medzi smerovačmi vytvorilo susedstvo a preniesli sa smerovacie informácie. Jeden simulačný krok predstavuje vykonanie jedného prechodu v modeli. Toto kritérium by sa dalo považovať za zovšeobecnenie kritéria času, keďže nie je závislé od konkrétnej implementácie modelu, ale iba od vykonávania samotného modelu. Keďže modely všetkých štyroch tried boli vytvorené tak, aby odzrkadľovali jednotlivé významné odlišnosti jednotlivých tried a nevyhnutné variácie služieb podporujúcich vytvorenie susedstva a výmenu smerovacej informácie a zvyšné časti modelu boli zhodné, môžeme kritérium počtu simulačných krokov považovať za vhodné a aj dostatočne všeobecné pre porovnanie modelov a následné stanovenie odporúčaní pre dizajn integrovaných smerovacích protokolov.

Keďže vybraté kritérium porovnania (počet simulačných krokov) závisí od spôsobu vykonávania modelu, podmieňujúcim krokom bolo vykonanie verifikácie jednotlivých modelov, ktorá bola popísaná v kapitole . Výpočtom stavového priestoru sme zistili, že vykonávanie niektorých z modelov môže skončiť v rôznych uzloch stavového priestoru, predpokladali sme, že aj počet simulačných krokov sa môže pri rôznych simulačných behoch jedného modelu líšiť. Preto sme sa rozhodli vykonať väčší počet replikácii simulačného behu (v našom prípade 1000 replikácii). Nástroj CPN Tools umožňuje spustiť viaceré replikácie automatickej simulácie pomocou definície:

```
CPN'Replications.nreplications 1000
```

pričom výsledkom je textový súbor s výpisom informácii o každom simulačnom behu. Príklad časti výstupu simulácie prvých dvoch replikácii modelu prvej triedy je nasledovný:

```
Simulation no.: 1
Steps.........: 180
Model time....: 0
Stop reason...: No more enabled transitions!
Time to run simulation: 0 seconds

Simulation no.: 2
Steps.........: 180
Model time....: 0
Stop reason...: No more enabled transitions!
Time to run simulation: 0 seconds
```

Dôležité položky výstupu každého simulačného behu sú číslo simulácie (*Simulation no.*), počet krokov, po vykonaní ktorých sa simulácia ukončila (*Steps*) a dôvod ukončenia simulácie (*Stop reason*). Keďže behov simulácie jednotlivých modelov je väčší počet, aby ich bolo možné súhrnne porovnať, rozhodli sme sa použiť štatistické funkcie aritmetický priemer a modus aby bolo z výslednej štatistiky zrejmé aký bol najčastejší počet krokov simulácie ako aj vplyv všetkých behov na výsledný počet simulačných krokov. Výsledky sú zosumarizované v tabuľke (Tabuľka 3).

	Modus	**Aritmetický priemer**
Simulácie modelu prvej triedy	180	179,98
Simulácie modelu druhej triedy	90	90
Simulácie modelu tretej triedy	252	252
Simulácie modelu štvrtej triedy	131	131

Tabuľka 3: Výsledky simulačných behov modelov tried riešenia protokolovej integrovanosti

Ako môžeme vidieť z výsledkov v tabuľke, počet krokov všetkých simulačných behov bol zhodný v prípade druhej, tretej a štvrtej triedy, čo indikuje rovnaký medián aj aritmetický priemer. Naopak simulačné behy modelu prvej triedy pozostávali buď zo 180 alebo 175 krokov, pričom 180 krokov sa vyskytovalo najčastejšie.

Z tabuľky taktiež vyplýva, že najlepšou triedou riešenia protokolovej integrovanosti na základe porovnania podľa počtu simulačných krokov je druhá trieda, a teda vytváranie jedného susedstva spolu s posielaním smerovacích správ iba na linkovej vrstve.

9 Odporúčania pre dizajn integrovaných smerovacích protokolov

Hlavným cieľom tejto dizertačnej práce, ktorý bol vytýčený v kapitole 5 je vytvorenie odporúčaní pre dizajn budúcich, alebo úpravu existujúcich integrovaných smerovacích protokolov. Preto sme vytvorili modely štyroch identifikovaných tried riešenia protokolovej integrovanosti z pohľadu počtu susedstiev a protokolu využitého na transport smerovacích správ, ktoré boli popísané v kapitole 6. Na základe verifikácie jednotlivých modelov a následného porovnania pomocou kritéria počtu simulačných krokov sme zistili, že najvhodnejšou triedou riešenia protokolovej integrovanosti je trieda dva.

Preto na základe výskumu prezentovaného v tejto dizertačnej práci je možné sformulovať nasledujúce odporúčania pre dizajn integrovaných smerovacích protokolov z pohľadu počtu susedstiev a transportu:

Smerovacie protokoly, ktoré sú navrhované pre použitie v sieťach, kde sa využívajú viaceré sieťové protokoly, teda integrované smerovacie protokoly, by mali vytvárať iba jedno spoločné susedstvo pre všetky použité sieťové protokoly. Taktiež na prenos správ smerovacieho protokolu by mali využívať len protokoly linkovej vrstvy a teda enkapsulovať smerovacie správy priamo do rámcov linkovej vrstvy. Ako základ smerovacích správ je vhodné využiť TLV štruktúry, ktoré umožňujú veľkú flexibilitu pre budúce rozšírenie samotného protokolu. Potvrdzovanie prijatia je potrebné riešiť v réžii smerovacieho protokolu, keďže linková vrstva takéto služby neposkytuje.

Ako ďalší z prínosov modelu druhej triedy vidíme jeho jednoduchosť čo potvrdil aj výpočet stavového priestoru v kapitole 7, keďže stavový priestor bol najmenší z modelov všetkých tried riešenia protokolovej integrovanosti. Keďže je model dostatočne jednoduchý, môže byť po pridaní vhodných rozširujúcich prvkov použitý aj na verifikáciu dizajnu konkrétnej implementácie smerovacieho protokolu založenej na formulovaných odporúčaniach po prečítaní popisu postupu vytvárania ako aj funkcionalít modelov jednotlivých tried v kapitole 6. Relatívne malý stavový priestor vytvoreného modelu môže potenciálne signalizovať výhodu tohto modelu aj v prípade porovnávania na základe iných kritérií alebo iných častí smerovacieho protokolu navrhnutého na základe týchto odporúčaní. Túto hypotézu by ale bolo potrebné overiť ďalším výskumom.

Výhodou modelu druhej triedy je aj fakt, že nie je potrebné riešiť problém, pomocou ktorého protokolu sa má vytvoriť susedstvo ako v prípade štvrtej triedy, popísanej v kapitole 6.2.1. Aj napriek tomu, že oba modely vytvárajú iba jedno susedstvo, keďže model druhej triedy používa na prenos smerovacích správ protokol linkovej vrstvy, podporované sieťové protokoly je dostatočné indikovať parametrom v hlavičke smerovacej správy. Grafickú reprezentáciu celého modelu druhej triedy riešenia protokolovej integrovanosti je možné nájsť v prílohách D a E tejto dizertačnej práce.

Keď sa pozrieme na aktuálne využívané integrované smerovacie protokoly, ako bolo uvedené v kapitole 1.4.3, jediným protokolom, ktorého dizajn zapadá do druhej triedy riešenia protokolovej integrovanosti je IS-IS, ktorý ale paradoxne nebol pôvodne vyvinutý pre IP siete. Jeho dizajn, nezávislý od použitých protokolov, mu umožnil byť adaptovaný aj v úplne iných typoch sietí, pre aké bol navrhnutý.

Keďže tretia trieda riešenia protokolovej integrovanosti vyšla z porovnania najhoršie, súčasné integrované smerovacie protokoly, ktoré zaraďujeme na základe ich vlastností do tejto triedy by mohli ťažiť z adaptovania vlastností druhej triedy protokolovej integrovanosti. Preto by cieľom ďalšieho výskumu mohlo byť napríklad prepracovanie protokolov ako sú EIGRP a OSPFv3 do druhej triedy protokolovej integrovanosti a následné porovnanie prepracovaného protokolu s originálnou predlohou.

Záver

Objavovanie sa nových trendov v oblasti počítačových sietí, ako aj používanie smerovacích protokolov v oblastiach iných, než sú klasické počítačové siete, vyvoláva množstvo otázok z hľadiska dizajnu smerovacích protokolov. Náš záujem upútala hlavne otázka podpory viacerých sieťových protokolov v smerovacom protokole a spôsob riešenia transportu na prenos smerovacích informácií. Keďže má naša katedra skúsenosti s vývojom otvorenej implementácie smerovacieho protokolu EIGRP, najväčšia inšpirácia pochádza práve z návrhu tohto protokolu ako aj z iných unikátnych smerovacích protokolov ako IS-IS.

V úvode práce sme načrtli dôležitosť smerovacích protokolov, ich význam a zdôraznili sme fakt, že smerovacie protokoly aj napriek tomu, že sú pre bežného používateľa neviditeľné, zabezpečujú dôležitú úlohu v riadení sieťovej prevádzky v prostredí internetu ako aj vo firemných, operátorských a iných rozľahlých sieťach.

V prvej kapitole sme sa venovali popisu princípu smerovania paketov v počítačových sieťach. Potom sme popísali princíp statických smerovacích záznamov, ich výhody a oblasť použitia. Tieto informácie sme rozšírili o poznatky z činnosti dynamických smerovacích protokolov. Následne sme popísali vlastnosti konkrétnych dynamických smerovacích protokolov s prihliadnutím na ich unikátne vlastnosti a ako bol ich dizajn ovplyvnený v prípade viacerých adresových rodín.

V druhej kapitole sme popísali služby nevyhnutné pre nadviazanie susedstva a posielanie smerovacích informácií vzhľadom na smerovacie protokoly podporujúce viaceré sieťové protokoly, ako aj identifikované triedy riešenia protokolovej integrovanosti.

V tretej kapitole boli popísané nástroje na vytvárane formálnych popisov systémov, ich výhody a nedostatky. Veľké množstvo zdrojov uvádzalo ako vhodný nástroj na modelovanie a následnú verifikáciu a analýzu sieťových protokolov, konkrétne aj smerovacích protokolov, farbené Petriho siete, ku ktorým sme uviedli aj príklad grafického znázornenia jednoduchého modelu.

Celá štvrtá kapitola je venovaná detailnému popisu smerovacieho protokolu EIGRP, jeho vlastnostiam popísaným v RFC špecifikácii ako aj funkciám, ktoré v tejto špecifikácii chýbajú. Následne sú porovnané dostupné otvorené implementácie smerovacieho protokolu EIGRP voči RFC. V poslednej časti druhej kapitoly sme sa zamerali na testovanie

popísaných funkcií otvorených implementácií v spoločnej virtualizovanej topológii. EIGRP bol venovaný v práci väčší priestor pre jeho výber ako základ pre modelovanie tretej triedy protokolovej integrovanosti.

Piata kapitola popisuje ciele dizertačnej práce, prečo sú dôležité a vyčleňuje čiastkové ciele potrebné pre splnenie načrtnutého cieľa celej práce.

Šiesta kapitola sa venuje jednotlivým modelom tried riešenia protokolovej integrovanosti vytvoreným pomocou farbených Petriho sietí a nástroja CPN Tools. Detailne popisuje prvky, z ktorých sa modely skladajú, načrtáva ich funkčnosť a popisuje problémy, ktoré bolo potrebné vyriešiť pre dosiahnutie požadovanej funkčnosti jednotlivých modelov tried.

V siedmej kapitole sme načrtli problémy verifikácie vytvorených modelov, postup identifikácie zdroja problémov a ich riešenie. V neposlednom rade siedma kapitola obsahuje výsledky verifikácie jednotlivých modelov.

Ôsma kapitola pojednáva o rôznych kritériách, ktoré je možné použiť na porovnanie viacerých modelov a vyberá najvhodnejšie z nich, ktoré boli zvolené na vytvorenie odporúčaní. V poslednej deviatej kapitole je predložená metodika pre dizajn integrovaných smerovacích protokolov na základe výsledkov porovnania jednotlivých modelov tried riešenia protokolovej integrovanosti a sú v nej popísané ďalšie výhody najlepšieho z modelov.

Ako ďalšie možnosti rozšírenia a budúceho výskumu odvíjajúceho sa od výsledkov tejto práce vidíme možnosť aplikácie odporúčaní pre dizajn integrovaných smerovacích protokolov do zmeny dizajnu súčasných smerovacích protokolov patriacich do tretej triedy protokolovej integrovanosti ako sú EIGRP a OSPFv3. Po zmene ich dizajnu a následnej implementácii by bolo možné porovnať prínosy odporúčaní stanovených na základe výskumu v tejto dizertačnej práci na reálnej implementácii smerovacieho protokolu. Taktiež je možné využiť postupy a princípy načrtnuté a preverené v tejto práci na vylepšenie iných častí integrovaných smerovacích protokolov.

Zoznam použitej literatúry

[1] Cisco Networking Academy, *Routing protocols companion guide*. Cisco Press, 2014.

[2] R. Graziani and A. Johnson, *Routing Protocols and concepts, CCNA exploration companion guide*. Cisco Press, 2007.

[3] N. Kocharians and P. Palúch, *CCIE Routing and Switching v5.0 Official Cert Guide, Volume 1*, 5th ed. Indianapolis: Cisco Press, 2014.

[4] K. Wallace, *CCNP Routing and Switching ROUTE 300-101 Official Cert Guide*. Indianapolis: Cisco Press, 2015.

[5] N. Kocharians, *CCIE Routing and Switching v5.1 Foundations: Bridging the Gap Between CCNP and CCIE*, 1st ed. Cisco Press, 2017.

[6] "CIDR Report." [Online]. Available: https://www.cidr-report.org/as2.0/. [Accessed: 01-Aug-2018].

[7] N. Kocharians and T. Vinson, *CCIE Routing and Switching v5.0 Official Cert Guide , Volume 2*. 2015.

[8] A. Lindem, S. Mirtorabi, A. Roy, M. Barnes, and R. Aggarwal, "Support of Address Families in OSPFv3," 2010.

[9] R. Callon, "Use of OSI IS-IS for routing in TCP/IP and dual environments," 1990.

[10] S. Aly, K. M.-D. U. July, and U. 2003, "Protocol verification and analysis using Colored Petri Nets," *Citeseer*, 2003.

[11] S. Šimoňák, "Modelovanie a analýza systémov s využitím integrácie formálnych metód [Habilitačná práca]," Technická univerzita v Košiciach, 2018.

[12] K. Jensen and L. M. Kristensen, *Coloured Petri Nets: Modelling and Validation of Concurrent Systems*. Berlin, Heidelberg: Springer Berlin Heidelberg, 2009.

[13] G. J. Holzmann and American Telephone and Telegraph Company., *Design and validation of computer protocols*. Prentice Hall, 1991.

[14] "Carl Adam Petri • IEEE Computer Society." [Online]. Available: https://www.computer.org/web/awards/pioneer-carl-petri. [Accessed: 01-Aug-2018].

[15] G. Hareesh and S. Chinara, "Dynamic Modeling of Routing Protocol Using Colored

Petri Nets," p. 47, 2015.

[16] D. Li, Y. Cui, K. Xu, and J. Wu, "Improvement of Multicast Routing Protocol Using Petri Nets," pp. 634–643, 2005, doi: 10.1007/11548706_67.

[17] D. A. Zaitsev, "Verification of Protocol BGP via Decomposition of Petri Net Model into Functional Subnets," *Proc. Des. Anal. Simul. Distrib. Syst. Symp.*, pp. 5–7, 2005.

[18] H. Chen, C. Zhou, Y. Qin, A. Vandenberg, A. V. Vasilakos, and N. Xiong, "Petri Net Modeling of the Reconfigurable Protocol Stack for Cloud Computing Control Systems," in *2010 IEEE Second International Conference on Cloud Computing Technology and Science*, 2010, pp. 393–400, doi: 10.1109/CloudCom.2010.81.

[19] D. Losch and J. RoBmann, "Visual Programming and Development of Manufacturing Processes Based on Hierarchical Petri Nets," in *2016 3rd International Conference on Soft Computing & Machine Intelligence (ISCMI)*, 2016, pp. 154–158, doi: 10.1109/ISCMI.2016.12.

[20] M. Naedele and J. W. Janneck, "Design patterns in Petri net system modeling," in *Proceedings. Fourth IEEE International Conference on Engineering of Complex Computer Systems (Cat. No.98EX193)*, 1998, pp. 47–54, doi: 10.1109/ICECCS.1998.706655.

[21] C. Seatzu, "Modeling, analysis, and control of automated manufacturing systems using Petri nets," in *2019 24th IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 2019, pp. 27–30, doi: 10.1109/ETFA.2019.8869012.

[22] Zhicao Dong and Xinhui Zhao, "Application of Petri Nets in Material Supply Systems," in *2010 Fourth International Conference on Genetic and Evolutionary Computing*, 2010, pp. 63–66, doi: 10.1109/ICGEC.2010.24.

[23] S. Niari, A. J.-P. of the 6th I. Conference, and U. 2013, "Verification of OSPF vulnerabilities by colored Petri net," *Proc. 6th Int. Conf. Secur. Inf. Networks*, pp. 102–109, 2013.

[24] D. Ibrahim, E. Sallam, … T. E.-… C. on C., and U. 2014, "Coloured petri net model for vector-based forwarding routing protocol," *Proc. Int. Conf. Comput. Technol. Inf. Manag.*, 2014.

[25] Z. J. Wang, J. Yang, G. Xie, and Mingtian, "OSPFv3 protocol simulation with colored

Petri nets," in *International Conference on Communication Technology Proceedings, 2003. ICCT 2003.*, 2003, vol. 1, no. 60273070, doi: 10.1109/ICCT.2003.1209078.

[26] K. Jensen, L. M. Kristensen, and L. Wells, "Coloured Petri Nets and CPN Tools for modelling and validation of concurrent systems," *Int. J. Softw. Tools Technol. Transf.*, vol. 9, no. 3–4, pp. 213–254, May 2007, doi: 10.1007/s10009-007-0038-x.

[27] K. Farah, K. Chabir, and M. N. Abdelkrim, "Colored Petri nets for modeling of networked control systems," in *2019 19th International Conference on Sciences and Techniques of Automatic Control and Computer Engineering (STA)*, 2019, pp. 226–230, doi: 10.1109/STA.2019.8717215.

[28] I. S. Brito and J. P. Barros, "Coloured Petri net model of the bCMS system using CPN tools," in *2013 3rd International Workshop on Comparing Requirements Modeling Approaches (CMA@RE)*, 2013, pp. 7–12, doi: 10.1109/CMA-RE.2013.6664178.

[29] B. Mikolajczak and A. Singh, "TransCPN - Software Tool for Transformation of Colored Petri Nets," in *2009 Sixth International Conference on Information Technology: New Generations*, 2009, pp. 211–216, doi: 10.1109/ITNG.2009.211.

[30] E. Mirzaeian, S. Ghaderi Mojaveri, H. Motameni, and A. Farahi, "An optimized approach to generate object oriented software test case by Colored Petri Net," in *2010 2nd International Conference on Software Technology and Engineering*, 2010, doi: 10.1109/ICSTE.2010.5608812.

[31] "Colored Petri Nets group Website." [Online]. Available: https://cs.au.dk/cpnets/. [Accessed: 06-Oct-2019].

[32] "CPN Tools Website." [Online]. Available: http://cpntools.org/. [Accessed: 06-Oct-2019].

[33] M. Zhou, F. DiCesare, and A. A. Desrochers, "A top-down approach to systematic synthesis of Petri net models for manufacturing systems," in *Proceedings, 1989 International Conference on Robotics and Automation*, pp. 534–539, doi: 10.1109/ROBOT.1989.100041.

[34] H. T. Jung and S. H. Joo, "Transformation of an activity model into a Colored Petri Net model," in *Trendz in Information Sciences & Computing(TISC2010)*, 2010, pp. 32–37, doi: 10.1109/TISC.2010.5714602.

[35] Yazhou Chen, "Research on a Top-Down collaborative modeling system," in *2008*

12th International Conference on Computer Supported Cooperative Work in Design, 2008, pp. 171–176, doi: 10.1109/CSCWD.2008.4536976.

[36] Y. Yang, S. Zhang, and X. Cheng, "Process Modeling of Top-down Collaborative Assembly Design Based on Petri Net," in *2007 10th IEEE International Conference on Computer-Aided Design and Computer Graphics*, 2007, pp. 401–406, doi: 10.1109/CADCG.2007.4407916.

[37] L. Ferrarini and M. Trioni, "Modeling shared resources with generalized synchronization within a Petri net bottom-up approach," in *Proceedings of IECON'94 - 20th Annual Conference of IEEE Industrial Electronics*, vol. 2, pp. 1105–1110, doi: 10.1109/IECON.1994.397946.

[38] C. M. M. Junior, R. M. S. Julia, and S. Julia, "Modeling Recursive Search Algorithms by Means of Hierarchical Colored Petri Nets and CPN Tools," in *2015 12th International Conference on Information Technology - New Generations*, 2015, pp. 788–791, doi: 10.1109/ITNG.2015.143.

[39] "Enhanced Interior Gateway Routing Protocol (EIGRP) Informational RFC Frequently Asked Questions - Cisco." [Online]. Available: https://www.cisco.com/c/en/us/products/collateral/ios-nx-os-software/enhanced-interior-gateway-routing-protocol-eigrp/qa_C67-726299.html. [Accessed: 26-Aug-2018].

[40] J. Expósito, V. Trujillo, and E. Gamess, "Easy-EIGRP: A Didactic Application for Teaching and Learning of the Enhanced Interior Gateway Routing Protocol," in *2010 Sixth International Conference on Networking and Services*, 2010, pp. 340–345, doi: 10.1109/ICNS.2010.53.

[41] D. Savage, J. Ng, S. Moore, D. Slice, P. Paluch, and R. White, "Cisco's Enhanced Interior Gateway Routing Protocol (EIGRP)," RFC Editor, 2016.

[42] J. J. Garcia-Luna-Aceves, "Loop-free routing using diffusing computations," *IEEE/ACM Trans. Netw.*, vol. 1, no. 1, pp. 130–141, 1993, doi: 10.1109/90.222913.

[43] R. Albrightson, J. J. Garcia-Luna-Aceves, and J. Boyle, "EIGRP - A fast routing protocl based on distance vectors," in *Proc. Networld/Interop*, 1994, vol. 94, pp. 136–147.

[44] P. Jakma and D. Lamparter, "Introduction to the quagga routing suite," *IEEE Netw.*,

vol. 28, no. 2, pp. 42–48, Mar. 2014, doi: 10.1109/MNET.2014.6786612.

[45] "Quagga Routing Suite Manual." [Online]. Available: http://www.nongnu.org/quagga/docs/docs-info.html. [Accessed: 01-Aug-2018].

[46] M. Kontšek, "Implementácia mechanizmov pre reštart susedských relácií do smerovacieho protokolu Quagga EIGRP [Diplomová práca]," University of Zilina, 2016.

[47] "Quagga-EIGRP source repository." [Online]. Available: https://github.com/janovic/Quagga-EIGRP. [Accessed: 01-Aug-2018].

[48] J. Janovic, "Implementácia smerovacieho protokolu EIGRP v balíku Quagga, transportná časť [Diplomová práca]," University of Zilina, 2015.

[49] M. Perina, "Implementácia smerovacieho protokolu EIGRP v balíku Quagga, časť DUAL [Diplomová práca]," University of Zilina, 2015.

[50] P. Orság, "Implementácia smerovacieho protokolu EIGRP v balíku Quagga, časť riadenia [Diplomová práca]," University of Zilina, 2015.

[51] T. Hvorkový, "Implementácia filtrovania obsahu smerovacej informácie do smerovacieho protokolu Quagga EIGRP [Diplomová práca]," University of Zilina, 2016.

[52] "FreeRangeRouting - A new Quagga fork with more open development." [Online]. Available: https://www.slideshare.net/apnic/freerangerouting-a-new-quagga-fork-with-more-open-development. [Accessed: 01-Aug-2018].

[53] "Welcoming FRRouting to The Linux Foundation." [Online]. Available: https://www.linux.com/news/2017/4/welcoming-frrouting-linux-foundation. [Accessed: 01-Aug-2018].

[54] "FRRouting source repository." [Online]. Available: https://github.com/FRRouting/frr. [Accessed: 01-Aug-2018].

[55] "EIGRP implementation for OpenBSD." [Online]. Available: https://github.com/rwestphal/openbsd-eigrpd. [Accessed: 01-Aug-2018].

[56] Y. He, T. Liu, H. Zhong, and Q. Xiong, "EA-CPNsim: A CPN-Based Simulation Platform for Analysis and Defense Design of Internet End-Systems Targeted Attacks," in *2009 Fourth International Conference on Frontier of Computer Science*

and Technology, 2009, pp. 548–552, doi: 10.1109/FCST.2009.68.

[57] F. Yutao, S. Guiping, H. Liu, and Z. Siyu, "Study on a CPN-Based Auto-Analysis Tool for Security Protocols," in *2012 Fourth International Symposium on Information Science and Engineering*, 2012, pp. 179–182, doi: 10.1109/ISISE.2012.45.

[58] T. R. Shmeleva, "Measuring subnets of colored Petri net models: Online technique for networks perfomance evaluation," in *2017 International Conference on Information and Telecommunication Technologies and Radio Electronics (UkrMiCo)*, 2017, pp. 1–5, doi: 10.1109/UkrMiCo.2017.8095417.

[59] Z. Abderrazzak and E. Ahmed, "Cloud SaaS using MDA approach on a multiview models generate a SaaS from a colored Petri Net using view PNML," in *2015 International Conference on Cloud Technologies and Applications (CloudTech)*, 2015, pp. 1–5, doi: 10.1109/CloudTech.2015.7336977.

[60] S. H. Askari, S. A. Khan, M. Haris, and M. Shoaib, "Pattern Based Model Reuse Using Colored Petri Nets," in *2019 19th International Conference on Computational Science and Its Applications (ICCSA)*, 2019, pp. 32–38, doi: 10.1109/ICCSA.2019.000-7.

10 Prílohy

Príloha A CD médium – dizertačná práca v elektronickej podobe a prílohy

Príloha B Model 1. triedy riešenia protokolovej integrovanosti – 1.časť

Príloha C Model 1. triedy riešenia protokolovej integrovanosti – 2.časť

Príloha D Model 2. triedy riešenia protokolovej integrovanosti – 1.časť

Príloha E Model 2. triedy riešenia protokolovej integrovanosti – 2.časť

Príloha F Model 3. triedy riešenia protokolovej integrovanosti – 1.časť

Príloha G Model 3. triedy riešenia protokolovej integrovanosti – 2.časť

Príloha H Model 4. triedy riešenia protokolovej integrovanosti – 1.časť

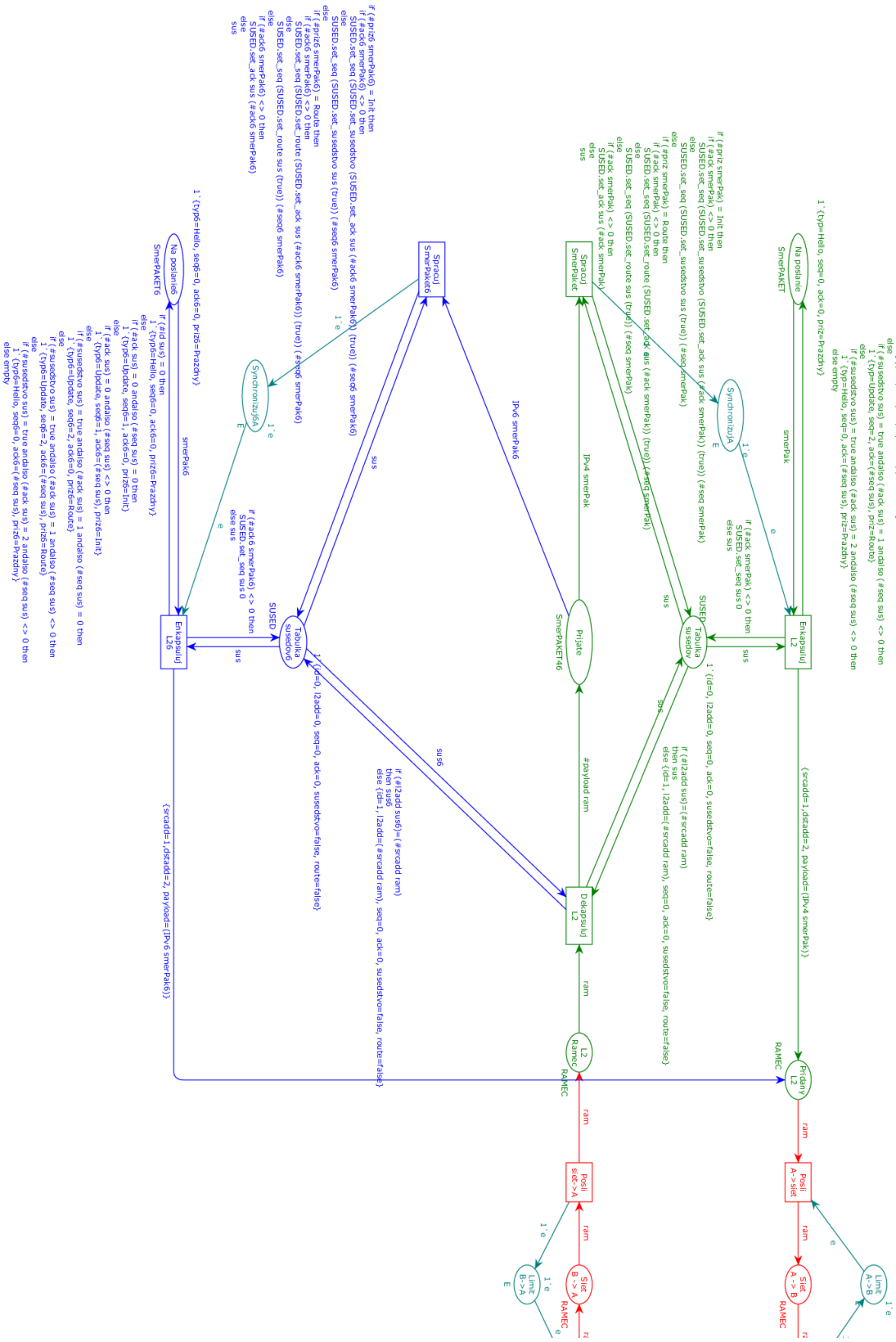
Príloha I Model 4. triedy riešenia protokolovej integrovanosti – 2.časť

Príloha A: Obsah CD

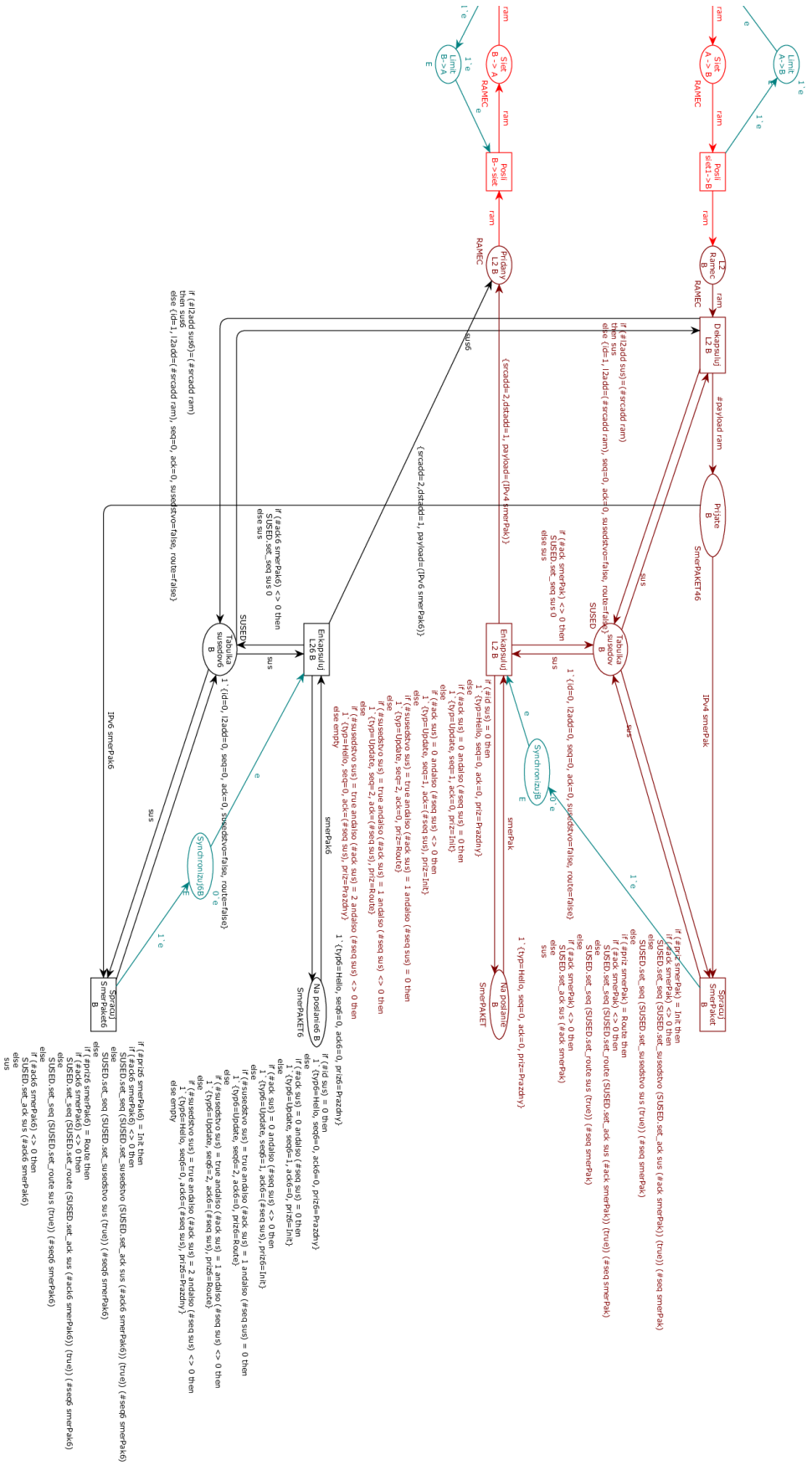
Súčasťou práce je CD obsahujúce dizertačnú prácu v elektronickej forme, zdrojové súbory jednotlivých modelov z programu CPN Tools, grafické reprezentácie jednotlivých modelov vo formáte PDF (vektorové) ako aj PNG a kompletné textové výpisy z verifikácie a simulačných behov jednotlivých modelov.

Zložka	Názov súboru	Popis
	DizertacnaPraca-Kontsek.pdf	Elektronická forma dizertačnej práce
modely	T1-L2-osobitne.cpn	Zdrojový súbor modelu prvej triedy
	T2-L2-spolu.cpn	Zdrojový súbor modelu druhej triedy
	T3-L3-osobitne.cpn	Zdrojový súbor modelu tretej triedy
	T4-L3-spolu.cpn	Zdrojový súbor modelu štvrtej triedy
obrazky-modelov		Zložka obsahuje grafické reprezentácie jednotlivých modelov vo formáte PDF aj PNG
verifikacia		Zložka obsahuje kompletné výpisy výpočtu stavového priestoru jednotlivých modelov
simulacia		Zložka obsahuje kompletné výpisy výsledku simulačných behov jednotlivých modelov

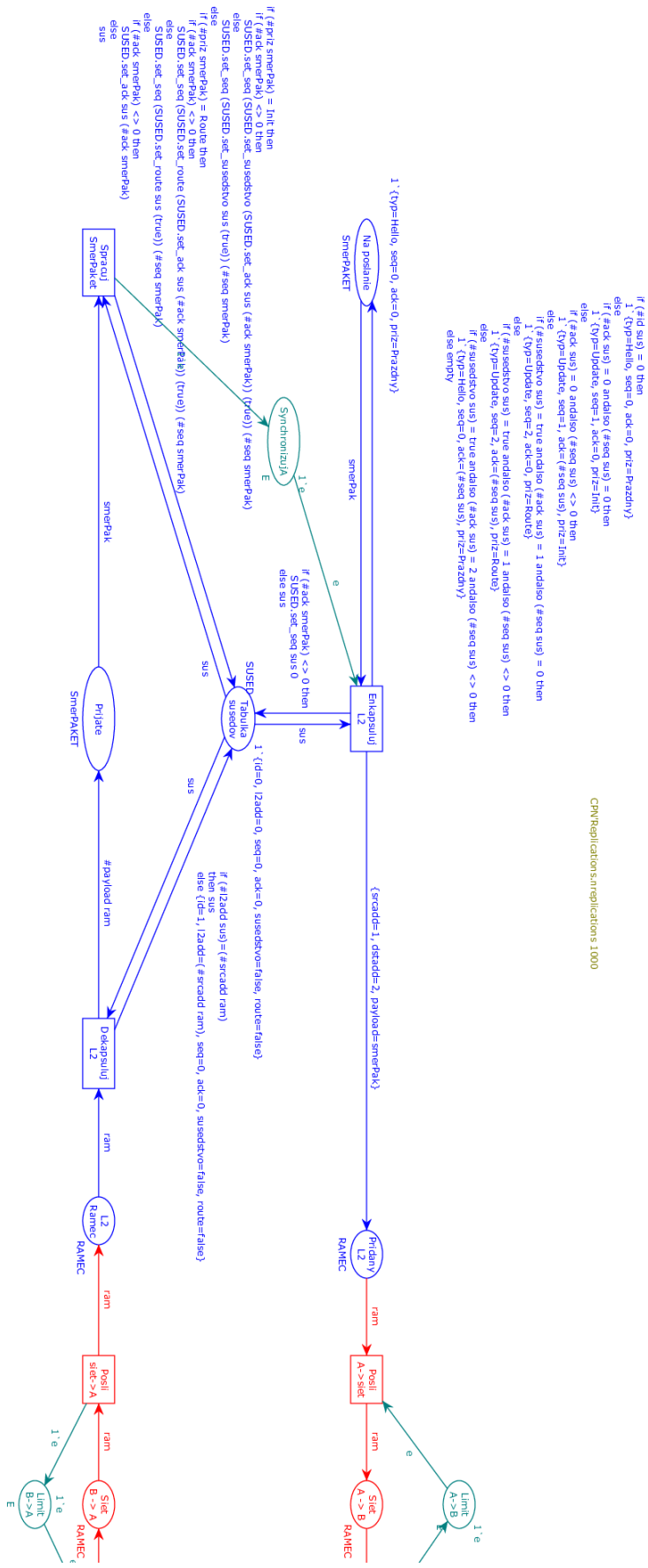
Príloha B: Model 1. triedy riešenia protokolovej integrovanosti – 1.časť



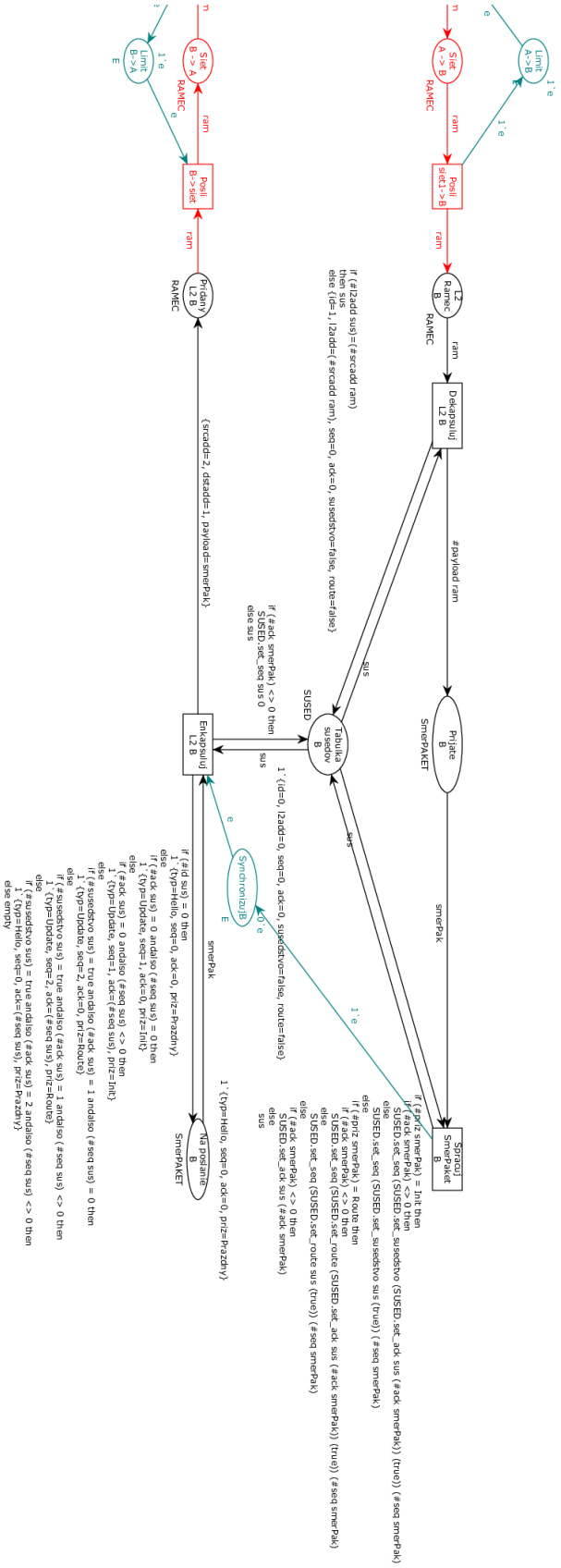
Príloha C: Model 1. triedy riešenia protokolovej integrovanosti – 2.časť



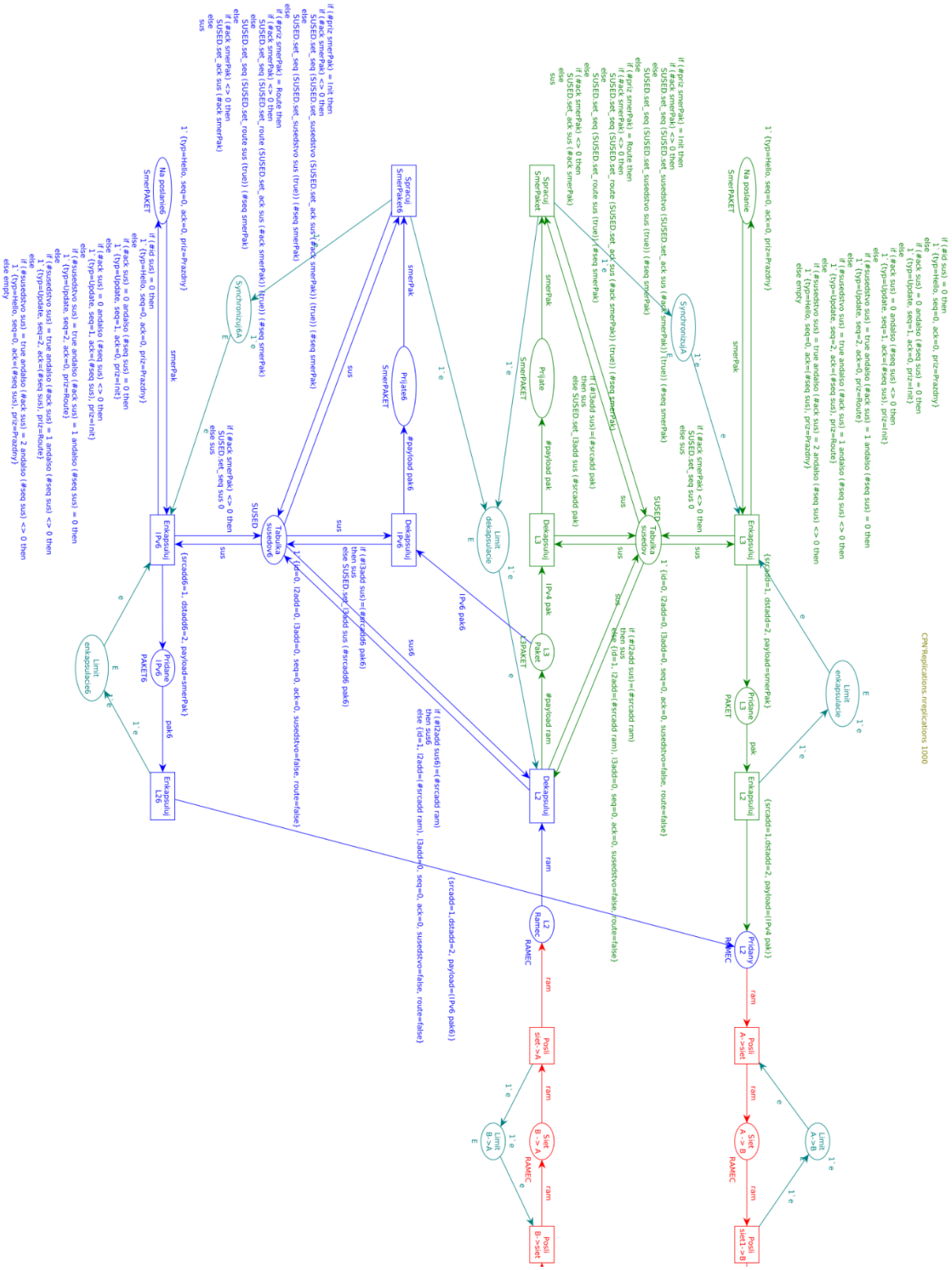
Príloha D: Model 2. triedy riešenia protokolovej integrovanosti – 1.časť



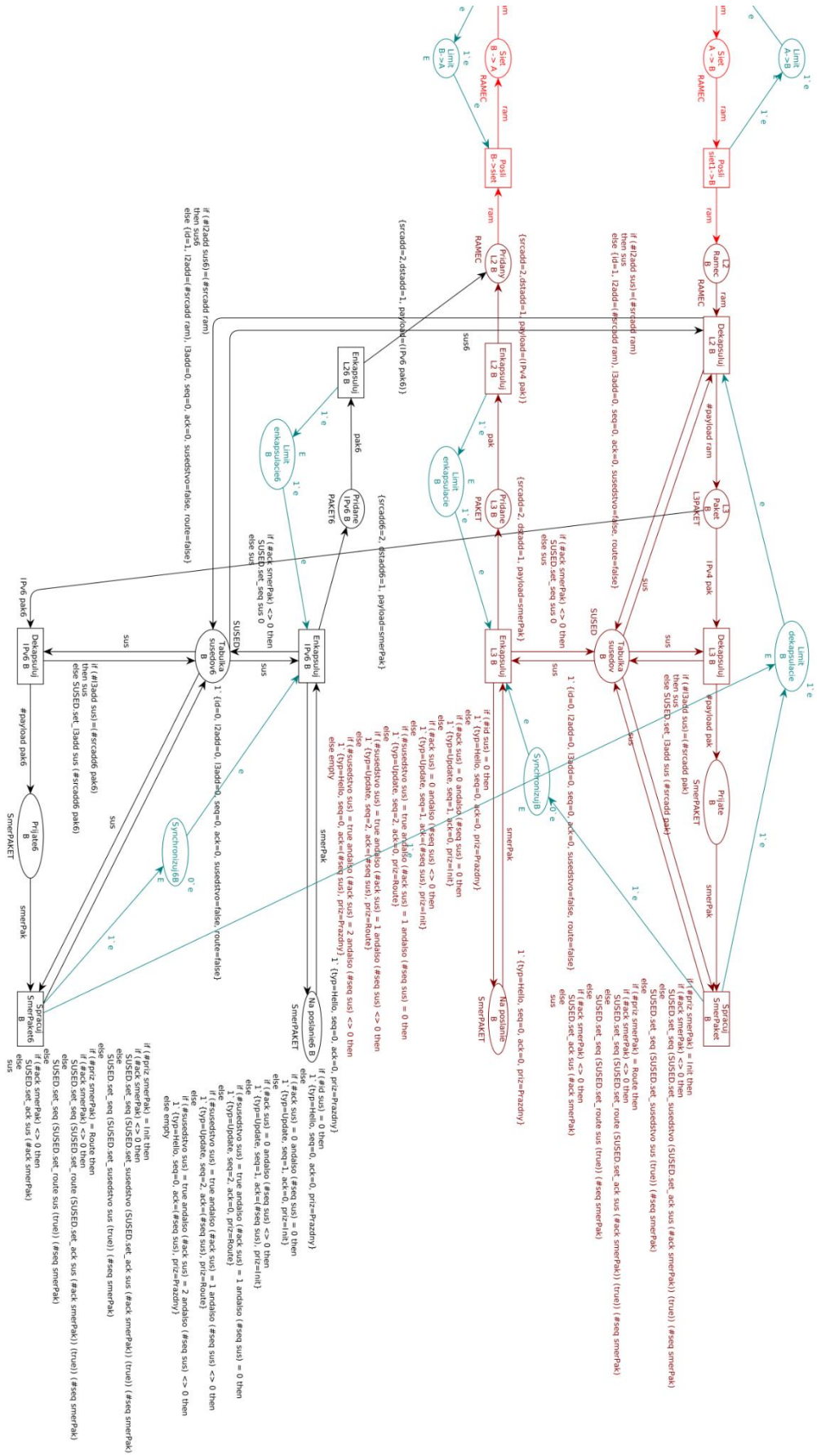
Príloha E: Model 2. triedy riešenia protokolovej integrovanosti – 2.časť



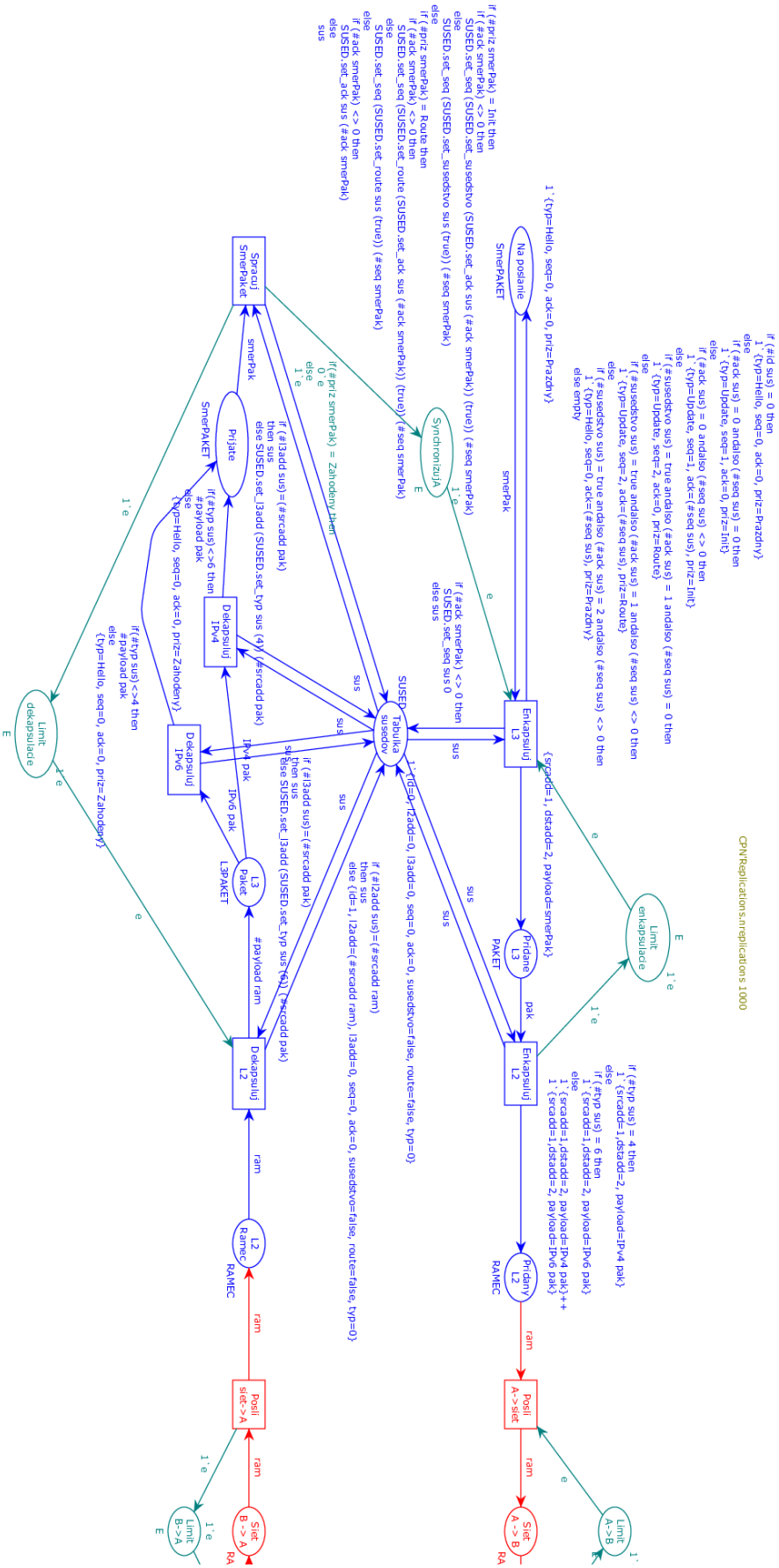
Príloha F: Model 3. triedy riešenia protokolovej integrovanosti – 1.časť



Príloha G: Model 3. triedy riešenia protokolovej integrovanosti – 2.časť



Príloha H: Model 4. triedy riešenia protokolovej integrovanosti – 1.časť



Príloha I: Model 4. triedy riešenia protokolovej integrovanosti – 2.časť