VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

KVALITA SLUŽBY V KONVERGOVANÝCH SYSTÉMECH S PRVKY ŘÍZENÝMI NEURONOVOU SÍTÍ

QUALITY OF SERVICE IN CONVERGED SYSTEMS WITH ELEMENTS CONTROLLED BY NEURAL NETWORK

DIZERTAČNÍ PRÁCE

DOCTORAL THESIS

AUTOR PRÁCE — Ing. MICHAL POLÍVKA

AUTHOR

VEDOUCÍ PRÁCE — doc. Ing. VLADISLAV ŠKORPIL, CSc.

SUPERVISOR

BRNO 2014

Abstrakt

Kvalita služby (QoS) je v konvergovaných systémech důležitým parametrem. Disertační práce se zabývá výzkumem její implementace do navrženého nového síťového prvku. Byl navržen a implementován nový protokol inspirovaný IP protokolem. V rámci řešení disertační práce byl navržen nový síťový prvek – přepínač vybavený řízením založeným na neuronové síti. V rámci naplňování cílů disertační práce byly zkoumány současné metody řízení přepínačů, několik přepínačů napříč výkonnostním spektrem bylo proměřeno. Na základě získaných poznatků byl navržen čtyřportový přepínač se spojovacím polem založeným na křížovém spínači s externím řízením. Přepínač byl navržen tak, aby v maximální možné míře podporoval QoS. Spojovací pole je řízeno neuronovou sítí typu „feedforward backpropagation". Navržený přepínač byl modelován v MATLABu a především v Simulinku. Provedené simulace prokázaly funkčnost navrženého řešení.

Abstract

The Quality of Service (QoS) is in converged systems an important parameter. The dissertation thesis deals with research of QoS implementation into a newly developed network element. There was designed and implemented new protocol, based on the IP. The dissertation thesis deals with proposal of a new network element – the switch controlled by a neural network. During the research have been measured switches cross a performance classes. On the base of the measurement was designed the new four-port switch with switch fabric build on crossbar switch with an external control. The switch was designed with maximum QoS support. The switch fabric is controlled by the feedforward backpropagation neural network. The designed switch was modeled in the MATLAB and Simulink. The simulations prove that developed solution is functional.

Klíčová slova

Přepínač, switch, neuronová síť, softwarové fronty, hardwarové fronty, MATLAB, Simulink, VNUML.

Keywords

Switch, neural network, software queues, hardware queues, MATLAB, Simulink, VNUML.

POLÍVKA, M. *Kvalita služby v konvergovaných systémech s prvky řízenými neuronovou sítí*. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2014. 120 s. Vedoucí dizertační práce doc. Ing. Vladislav Škorpil, CSc.

PROHLÁŠENÍ

Prohlašuji, že svou doktorskou práci na téma „Kvalita služby v konvergovaných systémech s prvky řízenými neuronovou sítí“ jsem vypracoval samostatně pod vedením vedoucího doktorské práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené doktorské práce dále prohlašuji, že v souvislosti s vytvořením této doktorské práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a/nebo majetkových a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon), ve znění pozdějších předpisů, včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č. 40/2009 Sb.

Brno 28. srpna 2014

PODĚKOVÁNÍ

Za možnost zúčastnit se mnoha výzkumů, přednášek a řešení řady projektů, mnohokrát za možnost „být u toho“, za řadu cenných rad a inspirativních myšlenek, za vytrvalou pomoc a podporu po celou dobu mého doktorského studia děkuji svému školiteli, panu doc. Ing. Vladislavu Škorpilovi, CSc.

Za pomoc a inspiraci ve druhém roce mého doktorského studia a nasměrování výzkumu novým směrem, za nový pohled na problematiku děkuji prof. Laurentu Perrotonovi z ESIEE Paris.

Za celoživotní podporu děkuji svým rodičům. Děkuji také všem zaměstnancům a studentům Ústavu telekomunikací za mnoho podněcujících otázek a diskusí.

SIX research centre
sensor, information and communication systems

Faculty of Electrical Engineering
and Communication

Brno University of Technology
Technicka 12, CZ-61600 Brno, Czechia

http://www.six.feec.vutbr.cz

Experimentální část této disertační práce byla realizována na výzkumné infrastruktuře vybudované v rámci projektu CZ.1.05/2.1.00/03.0072
Centrum senzorických, informačních a komunikačních systémů (SIX)
operačního programu Výzkum a vývoj pro inovace.

MINISTERSTVO ŠKOLSTVÍ, MLÁDEŽE A TĚLOVÝCHOVY

EVROPSKÁ UNIE
EVROPSKÝ FOND PRO REGIONÁLNÍ ROZVOJ
INVESTICE DO VAŠÍ BUDOUCNOSTI

2007–13
OP Výzkum a vývoj pro inovace

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

Úvod

Vytyčeným cílem předložené disertační práce je navrhnout architekturu nového síťového prvku zohledňujícího pravidla QoS v moderních konvergovaných sítích. Disertační práce volně navazuje na projekt GA ČR 102/07/1503 „Pokročilá optimalizace návrhu komunikačních systémů pomocí neuronových sítí“, který byl v roce 2009 úspěšně uzavřen.

Vytčeným cílem disertační práce je využít v návrhu architektury síťového prvku neuronových sítí. Neuronovou síť implementovat jako řídicí prvek. Síťovým prvkem se rozumí přepínač (switch) nebo směrovač (router).

Výzkum probíhající v rámci řešení disertační práce měl ambice navrhnout nový směr, který by mohl předestřít v oblasti síťové infrastruktury nové možnosti návrhu a řízení síťových prvků, jakými jsou mj. přepínače Cisco Nexus 9000, případně Catalyst 6800 nebo směrovač Cisco CRS-3, resp. Brocade MLX.

Záměr využít k řízení umělé neuronové sítě vychází především z výsledků zmíněného předchozího výzkumu. Použití neuronových sítí se osvědčilo v úlohách, kde je třeba rychle a efektivně navrhnout řešení komplikovaných problémů, při nichž nelze využít konvenční algoritmy.

Výkon a další parametry současných aktivních síťových prvků umožňuje pokrytí všech zásadních požadavků kladených na soudobé konvergované sítě. V některých případech se aktivní prvky obtížně vyrovnávají se stavy, kdy je síť přetížena provozem a pracuje na hranici své propustnosti. V zásadě ale platí, že problémy na sítích nejsou obvykle působeny nedostupností dostatečně výkonných síťových prvků, ale spíše chybami v návrhu nebo konfiguraci dostupných řešení.

Přenosová kapacita datových sítí roste podle Gilderova zákona třikrát rychleji než kapacita výpočetní (zákon formuloval Gregore Gilder ve své knize „Telecosm“ [1]), což znamená zdvojnásobení každých 6 až 8 měsíců. Podle „VNI Forecast Highlights“ [2] bude v roce 2018 měsíčně přeneseno přes Internet 102,2 EB dat oproti 36,4 EB v roce 2013. Podíl přenosu videa stoupne podle [2] z 66 % v roce 2013 na 79 % v roce 2018. Odhady v míře nárůstu přenosové kapacity se rozcházejí, ale samotný růst je trvalou příležitostí pro výzkum nových algoritmů a vývoj nových síťových prvků. Poptávka po nových prvcích je důsledkem tlaku z jedné strany na pokles cen dostupných řešení a ze strany druhé zvyšování požadavků na přenos větších objemů dat. Možnou cestou je odklon od tradiční architektury a otevření nových cest.

Soudobé konvergované sítě propojují koncepce původně oddělených telekomunikačních a datových sítí. Rozdíl mezi pojetím klasické telekomunikační a datové sítě postupně vymizel. Současné konvergované sítě slouží k přenosům jak hlasu, tak videa, televize atd. Zastřešujícím protokolem je protokol IP. Na straně přístupových sítí je u koncových zákazníků obvyklá pouze jediná přípojka k síti pokrývající všechny potřeby zákazníka počínaje telefonem, přes televizi a konče připojením k Internetu. Přípojky nabízené koncovým zákazníkům operátory se liší především prostředím, ze kterého operátoři vzešli – přípojka může být tedy primárně xDSL, která je typická pro původní telekomunikační operátory, koaxiální TV přípojka obvyklá u původních poskytovatelů kabelové TV. Případně jsou postupně budovány nové sítě lokálních i globálních operátorů poskytujících od počátku konvergované služby. Nosnou technologií je v tomto případě obvykle pasivní optická síť, bezdrátová síť nebo metalická strukturovaná kabeláž. Služby poskytované konvergovanými sítěmi zákazníkům jsou napříč poskytovateli téměř identické.

Při vývoji síťových prvků je jedním z inspirujících faktorů snaha dosáhnout zpoždění původních telekomunikačních sítí s přepojováním okruhů, která byla minimální a byla dána především fyzikálními limity rychlosti šíření signálu po přenosovém médiu. Zpoždění v datových sítích, nyní sítích konvergovaných, narůstá na každém aktivním prvku. Snahou výrobců je jeho minimalizace. Problematice zpoždění a míst jeho nárůstu v konvergovaných sítích se věnuje kapitola 1.1.

Skutečná kapacita (propustnost) transportních sítí komerčních operátorů není většinou zveřejňována. Určitou představu lze získat z velkoobchodních nabídek nebo ze statistik peeringových uzlů (v ČR t. č. NIX.CZ a Peering.cz). Vzhledem k propustnosti přístupové sítě zákaznických přípojek deklarované operátory a veřejně známým kapacitám transportních sítí a peeringových uzlů je zřejmé, že dosažení maximálního zatížení transportní sítě, resp. některých uzlů, není nereálné. Všechny datové sítě se budují s ohledem na ekonomiku provozu. Snahou operátorů je obsloužit maximum zákazníků prostřednictvím sítě s minimální přenosovou kapacitou, protože její růst zvyšuje náklady na její provoz.

Mechanismy QoS se v konvergovaných sítích uplatňují nejen ve stavech, kdy zatížení sítě dosáhne maxima propustnosti, ale také při běžném provozu. Nasazení mechanismů QoS i při běžném provozním zatížení je obvyklé jako prostředek dosažení co nejlepších parametrů sítě z pohledu zákazníka. Provoz přenášený historickou čistě analogovou telekomunikační sítí se neukládal ve vyrovnávacích pamětech, zpoždění v celém přenosovém řetězci bylo minimální. Koncepce konvergovaných sítí i jejich klíčových prvků je od původních telekomunikačních sítí odlišná. Datové sítě jsou levné a rychlost spojování je ve srovnání s klasickými telekomunikačními sítěmi velmi velká, ale připojená zařízení „soutěží“ o zdroje, tedy o možnost přenášet data.

V IP sítích, které jsou v různých modifikacích nejpoužívanější technologií transportních sítí, je přepínán, resp. spojován, každý jeden paket [3]. Spojení je ustanovováno pouze na přenesení tohoto jednoho paketu. Pokud v klasické telefonii trvá spojení řádově minuty, v případě datových sítí se jedná o několik řádů méně – dle rychlosti sítě a velikosti paketu, např. pro 100 Gb/s síť a 64B paket cca 5,12 ns.

Problematika QoS v IP sítích je shrnuta v kapitole 1, která kromě toho shrnuje dostupné zdroje literatury a celkově současný stav řešené problematiky.

1 PŘEHLED SOUČASNÉHO STAVU

Kapitola shrnuje současný stav oboru, dostupnou literaturu a pohled na řešenou problematiku. Snahou bylo sumarizovat zejména poznatky vztažené k aktuálně platným normám, doporučením RFC, doporučením výrobců a také manuálům na trhu dostupných aktivních prvků se vztahem k problematice QoS. Dále pak shrnout podporu QoS na různých vrstvách RM ISO/OSI (ISO 7498) [4] počínaje 2. vrstvou.

Pokud nebude uvedeno jinak, je veškerá popisovaná problematika vztažena k Ethernetu – IEEE 802.3 [5], ale často platí i pro ostatní paketové sítě.

Výrazné zaměření této kapitoly na aktivní prvky Cisco je dáno zejména jejich všeobecnou rozšířeností a možností fungování popsaných algoritmů prakticky ověřit. Při výzkumu byly využity zejména prvky Cisco 2821, Cisco Catalyst 6500, Catalyst 3560E a Catalyst 2960G, které jsou výkonnostním a hierarchickým průřezem typické soudobé sítě. Popsané mechanismy jsou ale obecně známé a používané většinou současných výrobců síťových prvků. Po analytické stránce jsou popsány detailně v [6].

1.1 POŽADAVKY SLUŽEB NA KVALITU SÍTĚ

Konvergovaná síť poskytuje svým uživatelům široké spektrum služeb. Řada zákazníků přitom používá specifické programy, protokoly nebo služby. Může se jednat o firemní informační systémy, zákaznicky řešené systémy pro on-line komunikaci, specifický software pro přenos hlasu, videa a další. Každá provozovaná služba má určité nároky na síť, po které je poskytována. Většina programů, jejichž úkolem je komunikovat po veřejné konvergované síti – Internetu, je pro svoji roli připravena a s proměnlivým prostředím počítá. Zohledňování takových služeb v rámci rozsáhlé sítě je ze strany operátora nepravděpodobné.

Odlišná je situace se službami nabízenými přímo operátorem koncovým zákazníkům, případně se službami poskytovanými v rámci firemní sítě. Za těchto podmínek jsou parametry a odlišné požadavky konkrétních protokolů, resp. služeb, v síti obvykle pečlivě analyzovány a zohledňovány přímo v návrhu struktury a konfigurace sítě.

Mezi typické služby se specifickými a současně dobře dokumentovanými požadavky patří telefonní služby poskytované po IP síti, tzv. VoIP (Voice over IP). Požadavky vybrané služby jsou obvykle popisovány potřebnou šířkou pásma, tedy minimální akceptovatelnou přenosovou rychlostí sítě a dále pak požadavky na další kvalitativní parametry – obvykle zpoždění, ztrátovost paketů a proměnlivé zpoždění (jitter), dále pak nepřímo ovlivnitelný požadavek na doručování paketů v pořadí, v jakém byly odeslány. Důsledkem nedodržení doporučených parametrů je zpravidla zhoršení použitelnosti, nebo je přímo provoz služby znemožněn.

Parametry sítě jsou ovlivňovány jak fyzickým médiem, resp. pasivními prvky sítě, tak aktivními síťovými prvky. Jejich správnou konfigurací lze užívání sítě zefektivnit.

1.1.1 POŽADAVKY VOIP NA KVALITU SÍTĚ

Telefonní provoz je jednou z prvních služeb, která je při přechodu na konvergovanou síťovou infrastrukturu začleňována. Zahrnutí VoIP služeb do sítě má přitom na její fungování nezanedbatelný vliv. Tab. 1.1 shrnuje přibližné požadavky na přenosovou rychlost v současnosti nejpoužívanějších VoIP kodeků [7]. Konkrétní zátěž závisí na nastavení konkrétního kodeku, charakteru přenášeného zvuku (signálu) atd.

Kodek	Datový tok 1 zařízení (kb/s)	Možných zařízení na 100Mb/s lince	Potřebná přenosová rychlost pro 10 zařízení (kb/s)
G.711	64,0	781	1 280
G.726r32	32,0	1 562	640
G.726r24	24,0	2 083	480
G.726r16	16,0	3 125	320
G.726	16,0	3 125	320
G.729	8,0	6 250	160
G.723r63	6,3	7 936	126
G.723r53	5,3	9 433	106

Tab. 1.1: Požadavky VoIP kodeků na šířku pásma [7]

Tab. 1.1 zachycuje pouze požadavky v jednom směru. Telefonní hovor je obvykle duplexní. Nároky VoIP na šířku pásma přenosové sítě lze různými prostředky snižovat – např. detekcí hlasové aktivity, použití kompresního protokolu cRTP (compressed RTP) atd. Tab. 1.1 nezahrnuje režii podpůrných protokolů a protokolů nižších vrstev. Některé z prostředků umožňující snížení zátěže síťové infrastruktury přímo souvisí s aktivními prvky použitými v síti. Příkladem je protokol cRTP (doporučení RFC 2508 [8], RFC 3544 [9] a RFC 3545 [10]), který slouží ke kompresi hlaviček IP, UDP a RTP protokolů.

Protokol RTP (Real-Time Transport Protocol) je spolu s RTCP (RTP Control Protocol) jedním z klíčových pro přenos streamovaného audia a videa sítí. Platná definice je v doporučení RFC 3550 [11] (aktualizace RFC 5506 [12], RFC 5761 [13], RFC 6051 [14], RFC 6222 [15]). RTP používá na transportní vrstvě UDP. Datová část RTP je velká obvykle 20–150 B [7], obvyklá hlavička protokolů IP, UDP a RTP je dohromady velká přibližně 60 B [7]. Z toho je patrná nízká efektivita protokolu RTP. Protokol cRTP byl navržen, aby neefektivitu protokolové sady IP – UDP – RTP odstranil. Nevýhodou použití cRTP je zátěž procesoru směrovače, na kterém je provozován. Proto je také doporučení [8], [9] a [10] definováno pouze pro sériové linky s nízkými rychlostmi (podle [7] do rychlosti 1,544 Mb/s). Na síti s nižšími přenosovými rychlostmi než je 1,544 Mb/s je použití protokolu cRTP doporučeno.

Kromě požadavku na propustnost sítě jsou doporučením ITU-T G.114 [16] stanoveny požadavky na maximální jednocestné zpoždění pro VoIP, ty jsou shrnuty v tab. 1.2.

Zpoždění [ms]	Popis
0–150	Akceptovatelné ve většině případů
150–400	Při vědomé implementaci technologie se zpožděním v tomto rozsahu stále akceptovatelné.
400<	Neakceptovatelné při projektování nové sítě.

Tab. 1.2: Maximální zpoždění VoIP podle ITU-T G.114 [7], [16]

Z tab. 1.2 vyplývá, že u nově budovaných konvergovaných sítí by podle doporučení G.114 celkové jednosměrné zpoždění nemělo přesáhnout 150 ms. Společnost Cisco Systems doporučuje dle [7] dodržet v síti VoIP zpoždění do 200–250 ms při minimalizaci jitteru (ideálně do 30 ms), viz kap. 1.1.3.

1.1.2 VZNIK ZPOŽDĚNÍ

Zpoždění, resp. latence, v konvergovaných sítích vzniká jako důsledek fyzikálních vlastností šíření signálu daným médiem – vzduchem, sklem, vodičem,… Toto zpoždění má na celkové

absolutní zpoždění, ke kterému v síti dochází, minimální vliv a pohybuje se v řádu µs, maximálně jednotek ms.

Zpoždění, které vnáší do přenosu aktivní prvky je výraznější. Zpoždění lze rozdělit na „pevné“ a „proměnlivé“ [7]. Proměnlivé zpoždění, viz kap. 1.1.3 – jeho vznik obvykle souvisí se sítí a frontami.

Pevné zpoždění má předpověditelný vývoj a je dáno především vlastnostmi síťových prvků. Může se jednat o dílčí zpoždění dané např. kódováním a dekódováním zvuku v rámci VoIP, paketizační zpoždění, případně zpoždění serializační atd. [7]. Obvyklým zdrojem zpoždění jsou fronty ve směrovačích a přepínačích. Nastavení front široce ovlivňuje vlastnosti aktivních prvků.

Zpoždění na každém aktivním prvku se sčítá. Celkové absolutní zpoždění lze vyjádřit rovnicí (1.1), kde (l_a) označuje absolutní zpoždění, (l_h) pevné zpoždění, (l_v) zpoždění proměnné a n je počet zdrojů zpoždění.

$$l_a = \sum_{i=0}^{n} \left(l_{h_i} + l_{v_i} \right) \tag{1.1}$$

Zpoždění vybraných aktivních prvků (přepínačů) bylo v rámci výzkumu měřeno, viz kap. 3.1.1. Změřená velikost zpoždění vkládaného přepínači se pohybovala v závislosti na velikosti paketů od 17,86 µs do 650,62 µs, viz tab. 3.1.

1.1.3 JITTER

Proměnlivé zpoždění je častěji označováno jako jitter [17]. Obvykle vzniká buď jako důsledek chyb v konfiguraci sítě nebo jako důsledek proměnlivých podmínek na síti. Jitter často výrazněji narůstá při přetížení sítě, případně jako důsledek chyby síťového prvku nebo jeho konfigurace – může se jednat o špatnou správu front, směrování paketů se stejným cílem různými cestami atd. Jitter je při přenosech VoIP, resp. obecně v konvergovaných sítích, nežádoucí. Jeho velikost se v síti pohybuje v rozmezí 30–50 ms [7].

Důsledkem jitteru přesahujícího doporučené meze, může být růst ztrátovosti paketů, které jsou na koncových stanicích a aktivních prvcích zahazovány kvůli doručení mimo pořadí. Jitter bývá aktivně potlačován nejčastěji na směrovačích pomocí vyrovnávací paměti, tzv. de-jitter buffer. Konsekventem potlačování jitteru je růst absolutního zpoždění [7], které je však většinou služeb lépe tolerováno.

1.1.4 ZTRÁTOVOST

Ztrátovost je dalším z parametrů, jež ovlivňuje QoS. V síti dochází ke ztrátovosti rámců/paketů z různých příčin – např. při přetečení vyrovnávacích pamětí. Ztrátovost je udávána v % provozu. Ztrátovost 100 % označuje ztrátu všech paketů. Příčinami ztrátovosti paketů mohou být [7]:

- Jitter přesahující meze dle doporučení – na síťových prvcích dochází k zahazování paketů doručených mimo pořadí nebo s velkým zpožděním, viz kap. 1.1.3.
- Nestabilní síť – dochází-li k výpadkům a opakovaným náběhům linek.
- Vady na fyzické vrstvě nemající za důsledek úplné vyřazení linky z provozu.
- Přetížení sítě. V síti pracující na hranicích propustnosti jsou zahazovány pakety, které není síť nebo daný konkrétní aktivní prvek schopen zpracovat.

Častou příčinou přetížení sítě je nedostatečný výkon aktivních prvků. Maximální propustnost aktivních prvků není dána rychlostí jejich rozhraní, ale vnitřní vybaveností a také službami na nich provozovanými. Např. směrovače Cisco 3900 jsou standardně osazovány 1 Gb/s porty a jsou schopny při všech zapnutých službách zpracovávat datový tok cca 350 Mb/s [18].

Za normálních podmínek je ztrátovost v síti 0%. V rámci výzkumu byla měřena reálná ztrátovost podle doporučení RFC 2544 na několika aktivních prvcích, viz kap. 3.1.4.

Tolerance ztrátovosti různými službami je různá. S tímto faktem pracují síťové prvky při uplatňování pravidel QoS. Tolerance ztrátovosti multimediálních služeb, mezi něž patří i VoIP, je obecně menší než v případě služeb, jako jsou přenosy souborů (FTP – File Transfer Protocol), HTML (HyperText Markup Language) stránek (HTTP – Hypertext Transfer Protocol) atp. VoIP systémy obvykle dokážou kompenzovat výpadek 20–50 ms provozu [7], ale s podmínkou ztráty pouze 1 paketu. Ztrátovost paketů by u VoIP neměla být větší než 1 % [7]. Při ztrátě více paketů se chyba projeví u uživatele (výpadkem zvuku během telefonátu).

Při přenosech multimédií, kam patří i VoIP, není efektivní přenášet ztracený paket opakovaně, protože při opakovaném zaslání by byl zahozen kvůli zpožděnému doručení. Opakované zaslání paketů má význam pouze u služeb, jakými jsou již zmíněné FTP, webové stránky aj.

V případě dalších služeb, jako je video na vyžádání – VOD (Video on Demand) a IPTV (Internet Protocol Television) – je tolerance ke ztrátovosti paketů často vyšší díky použitým kodekům. Tolerance jitteru a celkového zpoždění je rovněž vysoká díky možnosti ukládat doručené pakety do vyrovnávací paměti. V případě videokonferenčních systémů je kvůli nutné synchronizaci obrazu se zvukem a současně požadavku na interaktivitu shodnou s VoIP situace obdobná, jako u zmíněného VoIP. Kratší výpadek v přenosu videa (bez výpadku audia) ale obvykle nevede ke snížené srozumitelnosti komunikace.

1.1.5 JEDNOTKY – VÝKON AKTIVNÍCH PRVKŮ

Jednotkou vyjadřující přenosovou rychlost, šířku pásma (bandwidth), resp. propustnost aktivního prvku je bit za sekundu. Výkonnostní charakteristiky aktivního prvku lze popsat i dalšími, často používanými nenormalizovanými jednotkami. Tyto jednotky lze mezi sebou převádět. V praxi ale nevyjadřují totéž.

Přenosová rychlost obvykle udává teoretické maximální hodnoty dané použitou přenosovou technologií. Ostatní jednotky jsou obvykle vztaženy ke konkrétní velikosti rámce/paketu a udávají reálné výkony aktivních prvků za udaných podmínek.

Rámcová rychlost (frame rate): Jednotkou je rámec za sekundu (frame/second, resp. frame/s). Na druhé vrstvě (L2) RM ISO/OSI udává, kolik rámců je reálně schopen aktivní prvek zpracovat a předat ze vstupu na výstup. Rámcovou rychlost lze odvodit z přenosové rychlosti podle rovnice (1.2), kde v_r vyjadřuje rámcovou rychlost, v_p přenosovou rychlost a R velikost rámce [6]. Reálná rámcová rychlost závisí na návrhu aktivního prvku a výkonnosti všech jeho částí. Je také závislá na charakteristice zpracovávaného provozu. Ze vztahu (1.2) plyne, že rámcová rychlost s rostoucí velikostí rámce klesá. Podle (1.2) je vypočtená v_r 1000Base-T Ethernetu pro nejmenší (84B) rámec 1,488 Mframe/s, pro největší (1 538B) rámec pak 81,274 kframe/s [19]. Rámcová rychlost by se tedy měla pohybovat v rámci tohoto rozsahu.

$$v_r = \frac{v_p}{R} \tag{1.2}$$

Paketová rychlost (packet rate), v_{pk}, je obdobou rámcové rychlosti, na L3 RM ISO/OSI. V anglické literatuře je kromě jednotky (p/s) často používáno i označení *pps* (packet per second). Reálná paketová rychlost aktivního prvku se liší v závislosti na zpracovávaném protokolu, aktivních službách na daném prvku atd. Např. přepínač Cisco Catalyst 4 500 vybavený supervizorem 6E má deklarovanou propustnost 320 Gb/s, při zpracovávání IPv4 dokáže zpracovat 250 Mp/s, při zpracovávání IPv6 pouze 125 Mp/s [20]. Ačkoli byl IPv6 paket navržen tak, aby bylo jeho zpracování co nejefektivnější, některé aktivní prvky a jejich firmware není pro zpracování IPv6 optimalizován tak, jako pro IPv4.

Počet spojení za sekundu (connections per second), jednotka (c/s). Udává rychlost, resp. počet spojení, který je prvek schopen sestavit za 1 s. Sestavování spojení zajišťuje, pro jeho složitost, softwarová část architektury aktivního prvku. Samotné předávání paketů, které po sestavení spojení následuje, je naopak obvykle hardwarově akcelerované a prvek nějak zvlášť nezatěžuje. Dle [19] Cisco Catalyst 6500 sestaví přibližně 325 000 c/s.

Celkový počet souběžných spojení (maximum concurrent connections, *mcc*). Udává, kolik spojení je schopno zařízení současně spravovat. Nemusí se jednat o spojení aktivní, ale navázaná. Závisí mj. na velikosti paměti aktivního prvku. Dle [19] je v případě Cisco Catalyst 6500 *mcc* rovno cca 4 000 000.

Počet transakcí (transaction per second, Tr/s) udává počet akcí (transakcí) realizovaných za sekundu. Např. pro Cisco ACE XML Gateway dosahuje výkonu 30 000 Tr/s [19].

Aktivní prvek by měl být navržen tak, aby každý z jeho parametrů korespondoval s ostatními, aby se nemohlo stát, že se z některé vlastnosti stane zranitelnost, která povede např. k DoS (Denial of Service, odepření služby) útoku, např. otevřením většího počtu spojení, než kolik je schopen prvek obsloužit (legitimní uživatelé nebudou obslouženi) [19].

Výkon aktivních prvků je závislý na návrhu aktivního prvku, ale i aktivních službách. Stejný prvek s různou konfigurací může dosahovat v testech i dramaticky odlišných výsledků. I při stejné konfiguraci se mohou ve standardních provozních podmínkách výsledky aktivního prvku s různým provozem lišit. Např. v případě Ethernetu a IP, zpracovávají aktivní prvky v jednou okamžiku pakety/rámce různých velikostí atd.

1.2 Implementace QoS v konvenčních aktivních prvcích

V současné době je podpora QoS v aktivních síťových prvcích – směrovačích i přepínačích – zcela běžná. Rozdíly mezi různými řadami a výrobci jsou, v případě implementace podpory QoS, relativně malé a závislé spíše na určení daného zařízení. Odlišené jsou zvolené použité algoritmy a jejich nastavení, viz např. kap. 1.3. Významné rozdíly jsou také ve výkonnosti, stabilitě a implementaci dalších služeb aktivních prvků. Přepínání paketů v aktivních prvcích, jejichž dokumentaci dali jejich výrobci k dispozici, je řízeno konvenčními algoritmy. Tyto algoritmy jsou stabilní, prověřené, ale mají určitá omezení, viz kap. 1.3 a 1.4.

Implementací QoS do sítě se myslí zavedení takových mechanismů, které umožní jednotlivým datovým tokům přiřadit parametry umožňující jejich zpracování na směrovačích a přepínačích různým způsobem. Zejména se jedná o prioritu daného paketu, pravděpodobnost zahození a případně i náležitost k určitému datovému toku.

Požadavky na zavedení QoS jsou svázány s konvergovanými sítěmi, v nichž má každá provozovaná služba určité požadavky na zpoždění doručovaných paketů, proměnlivost zpož-

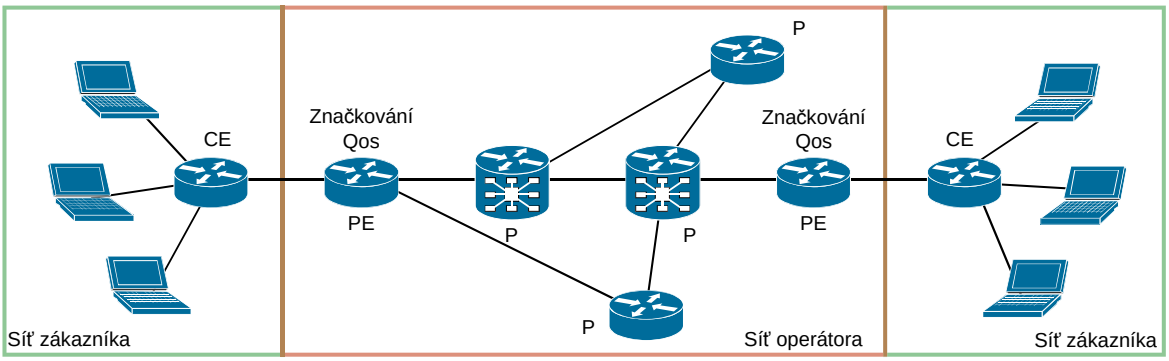
dění, chybovost či ztrátovost paketů a ty mohou být odlišné od ostatního přenášeného provozu. Mechanismy QoS lze zvolené toky upřednostňovat, případně zajistit, aby stanovenou šířku pásma nepřekročily.

Ve stavu přetížení sítě, ale i během normálního provozního režimu, mohou nastat situace, za kterých je nutné některé pakety na přepínačích a směrovačích zahazovat. Bez implementace QoS do sítě je s pakety nakládáno obvykle metodou „Best Effort“, do češtiny překládané jako „nejlepší úsilí“. Je-li nutné zahazování paketů, je důsledkem využívání algoritmu „Best Effort“ zahazování paketů napříč všemi zpracovávanými protokoly a pakety rovnoměrně bez jejich diverzifikace. Vzhledem k různým požadavkům různých služeb na parametry sítě, viz kap. 1.1, je toto řešení téměř vždy nevhodné.

Klasifikace a značkování (tedy přidání značky do hlavičky daného protokolu, viz kap. 1.2.5) provozů QoS je obvykle prováděno na hraničním směrovači sítě (PE – Provider Edge), nedoporučuje se povolit značkovat provoz na zákaznických směrovačích (CE – Customer Edge) [21]. Důvodem doporučení je snaha o zachování bezpečnosti sítě – pokud by bylo zákazníkovi umožněno provoz značkovat libovolně, mohl by snadno přetížit celou síť. Doporučení [21] tedy říká, že je vhodné značkovat provoz co nejblíže zákazníkovi, aby bylo možné předávat síťový provoz co nejplynuleji po transportní síti (P – Provider), ale nepřipustit, aby provoz označoval zákazník sám, obr. 1.1.

Odlišná je situace v případě tzv. interprovider peeringu, tedy při propojení sítí dvou vzájemně si důvěřujících subjektů. Na aktivních prvcích lze příchozí provoz přeznačkovat. Nakládání s určitým způsobem označkovaným provozem není závazné, s označkovanými pakety lze nakládat libovolným způsobem. Lze také různě zpracovávat pakety přicházející od různých „zákazníků“ nebo na různá rozhraní. Lze také respektovat jen vybrané značky vybraných protokolů – typicky VoIP. Uvnitř sítě je přípustné i značkování provozu přímo v důvěryhodném zdrojovém zařízení. Nejčastěji se jedná o videokonferenční systémy, IP telefony apod.

Označení zákazník a operátor je použito v kontextu terminologie zavedené např. v RFC 4026 [22]. V praxi se jedná o libovolné sítě s libovolným vztahem jejich provozovatelů s tím, že „zákaznická síť“ je hierarchicky níže – podřízena síti transportní, resp. operátorské.



Obr. 1.1: Značkování na PE směrovačích

Na obr. 1.1 je zachycen příklad propojení, topologie 2 sítí různých subjektů. Přenos pravidel QoS napříč propojenými sítěmi není častý. Obvyklejší je striktní oddělení a použití jednotných pravidel pouze v síti pod jednotnou správou.

Provoz bývá obvykle značkován nejčastěji na 3., síťové, vrstvě RM ISO/OSI, viz kap. 1.2.2 a 1.2.3. Méně obvyklé je značkování na 2. vrstvě, kap. 1.2.1.

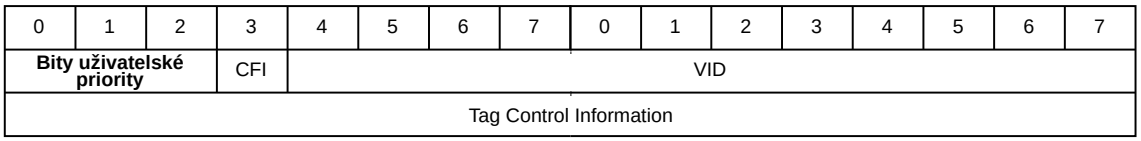
Při funkčnosti celého systému značkování, politik, správy front,… je důležité, aby bylo značkování a práce s označkovanými pakety konzistentní v celé síti. Je-li to možné, aby bylo QoS podporováno na všech aktivních prvcích sítě bez výjimek.

1.2.1 Ethernetový rámec a QoS

Značkování na úrovni 2. vrstvy RM ISO/OSI se používá obvykle pouze v případě, že přepínače použité v síti nejsou schopny pracovat s QoS na 3. vrstvě. Značkování je, vzhledem k charakteru 2. vrstvy, platné pouze do přechodu paketu mezi směrovači, tedy po dobu zachování ethernetové hlavičky. Standard Ethernetu – IEEE 802.3 [23] (první verze standardizována 1983, nyní aktuální verze 2012 [5]) neobsahuje pole „priorita“, které se ke značkování provozu na 2. vrstvě používá.

Pole „priorita“ zavádí doplňující standard (protokol) IEEE 802.1Q [24]. Vložená část hlavičky je dlouhá 2 B (2 oktety). 802.1Q pole „priorita“ nazývá PCP (Priority Code Point) [24]. Protokol 802.1Q vznikl primárně kvůli trunkingu VLAN (vzájemnému propojení virtuálních LAN sítí). Obdobnou úlohu má i protokol ISL (Inter-Switch Link) [25] navržený firmou Cisco. V praxi se používají obě řešení. Protokol ISL je z hlediska návrhu transparentnější než 802.1Q, protože zapouzdřuje původní ethernetový rámec. Naproti tomu 802.1Q do původního ethernetového rámce vkládá 2B pole označené ve standardu „VLAN Tag Control Information“ a do češtiny podle [21] překládáno jako „bity uživatelské priority“. Literatura [21] v případě 802.1Q hovoří o „značkování rámců“. Pole „Tag Control Information“ v ethernetové hlavičce následuje za MAC adresou zdroje, oddělená hodnotou 0x8100 [21].

Pro značení priority jsou v 802.1Q dedikovány 3 nejvýznamnější bity (na obr. 1.2 jsou to bity 0–2) vkládaného 2B pole Tag Control Information. Pole CFI (Canonical Format Indicator), následující za bity uživatelské priority, je jednobitové. Poslední 12b pole VID (VLAN identifier) slouží k identifikaci VLAN, hodnoty VID viz [24].

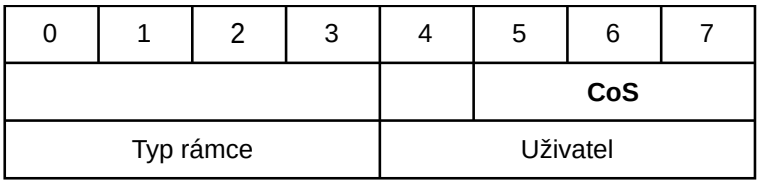


Obr. 1.2: Pole „Tag Control Information“ protokolu 802.1Q s vyznačením „bitů uživatelské priority“ [24]

Hodnoty pole „bity uživatelské priority“ stanovuje rovněž norma 802.1Q [24] ve své příloze G. Doporučené hodnoty priority, viz příloha A.1.

Protokol ISL navržený firmou Cisco zapouzdřuje původní ethernetovou hlavičku. Před původní ethernetovou hlavičku definovanou v 802.3 [23] přidá 26B hlavičku ISL, která kromě jiného obsahuje znovu odchozí a cílovou MAC adresu a také pole priority – označované jako „třída služeb“ CoS (Class of Service) [21]. Za původní ethernetový rámec je protokolem ISL umístěn nový kontrolní součet (FCS – Frame Check Sequence), přičemž původní kontrolní součet je odstraněn [21].

Hlavička protokolu ISL má strukturu od 802.1Q odlišnou. Částí hlavičky ISL je pole „uživatel“ (user), to je pro značkování provozu klíčové. Pole „uživatel“ je 4b a předchází mu 4b pole „typ rámce“, viz obr. 1.3. Dohromady tvoří jeden oktet. Pole „typ rámce“ nabývá v případě Ethernetu hodnoty 0 (v případě FDDI 2 atp., viz [25]). Pole „uživatel“ je 4b, ale aktuálně se používají pouze 3 nejméně významné bity (bity 5–7 na obr. 1.3) [21], [26]. Hodnoty pole CoS jsou kompatibilní s hodnotami pole „bity uživatelské priority“. Hodnoty pole CoS dané předpisem [25], viz příloha A.2.



Obr. 1.3: Pole „uživatel“ protokolu ISL s vyznačením pole CoS [25]

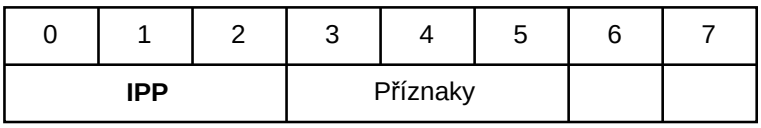
Většina konfigurovatelných směrovačů umí konvertovat hlavičku ISL na 802.1Q a naopak. V technické praxi je pole pro označení priority na 2. vrstvě RM ISO/OSI obvykle označováno jako CoS bez ohledu na to, zda se jedná o pole nacházející se v hlavičce ISL nebo 802.1Q. Jakkoli to není obvyklé, mohou provoz na 2. vrstvě značkovat i koncová zařízení, je pouze potřeba, aby osazená síťová karta podporovala trunking – tedy protokol ISL, případně 802.11Q. Značkování koncovými zařízeními na 2. vrstvě má význam pouze v lokální síti pod jednotnou správou, viz kap. 1.2.

1.2.2 Hlavička IPv4 a QoS

IP protokol verze 4 (IPv4) je postupně nahrazován moderním IP protokolem verze 6 (IPv6), přesto je IPv4 v současné době nejpoužívanějším protokolem síťové vrstvy. Hlavička protokolu IPv4 je definována doporučením RFC 791 [27] a aktualizována mj. doporučením RFC 1349 [28] (později nahrazeno RFC 2474 [29], viz dále). Pro potřeby QoS bylo v hlavičce IPv4 zmíněným doporučením RFC 791 definováno pole „typ služby“, ToS (Type of Service). V aktualizované verzi hlavičky pole „typ služby“ zůstalo, ale bylo novějšími doporučeními RFC 2598 [30], které bylo nahrazeno RFC 3246 [31] předefinováno.

Původní pole ToS bylo 1B, viz obr. 1.4. Pole ToS se v hlavičce nachází za 1. B, který je rezervován pro identifikaci verze IP protokolu a informaci o délce hlavičky. První (nejvýznamnější) 3 b jsou nazvány „IP priorita“, resp. IPP (IP Precedence), na obr. 1.4 bity 0–2. Pole příznaků následující za polem IPP je, podle RFC 791, 3b ([27], str. 12), na obr. 1.4 bity 3–5. V poli příznaků 3. b určoval charakter zpoždění, 4. propustnost a 5. spolehlivost, přičemž hodnota 0 označovala normální stav a hodnota 1 lepší podmínky. Bity 6–7 byly podle RFC 791 ponechány jako rezervní pro budoucí využití.

V technické praxi se prakticky využívalo pouze pole IPP, tedy první 3 b pole typ služby, zbývající pole se nevyužívala. Seznam priorit, rozlišovaný v IPP, viz příloha A.3, resp. [27].

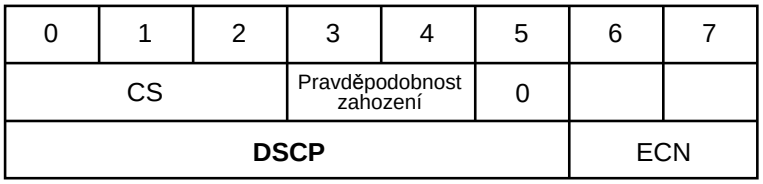


Obr. 1.4: Pole „typ služby“ (ToS) s vyznačením pole IPP, podle RFC 791 [27]

Pole ToS bylo v novějších RFC 3246 [31] a RFC 3168 [32] (aktualizované RFC 4301 [33] a RFC 6040 [34]) předefinováno a bylo označeno jako pole „diferencované služby“, DS (Differentiated Services, zkráceně DiffServ). Pole DS je pro protokol IPv4 i IPv6 přímo definováno v dokumentu RFC 2474 [29], doplňkové guidelines pak v RFC 2780 [35]. Architektura DiffServ je definována v RFC 2475 [36].

Prvních 6 b (na obr. 1.5 bity 0–5) pole DS je pole DSCP (Differentiated Services Code Point). V RFC 3168 [32] jsou definovány zbylé 2 nejnižší bity, z celkových 8 b, pole DS. Ty jsou pojmenovány ECN (Explicit Congestion Notification), [21] je pojmenovává česky jako „explicitní oznamování zahlcení“, Na obr. 1.5 pole ECN reprezentují bity 6 a 7. Pole DS se v IPv4 hlavičce nachází na stejné pozici, jako pole ToS.

Kvůli zachování zpětné kompatibility s původním polem ToS zachovávají první tři bity DSCP (bity 0–2 na obr. 1.5) svůj původní význam. Mechanismus zpětné kompatibility je zaveden dokumentem RFC 2475 [36], který zavádí tzv. Class Selector (CS), tedy selektor třídy [21]. Binární vyjádření bitů 0–2 (v obr. 1.5 označené CS) pole DSCP odpovídá binárnímu vyjádření priority polem IPP. Bity 3–4 (v obr. 1.5 označeny „pravděpodobnost zahození“) mohou být rovny 0 (potom má DSCP obdobné vlastnosti jako IPP) nebo od 0 různé, viz dále. Poslední bit (na obr. 1.5 bit 5) pole DSCP je podle současných RFC roven vždy 0.



Obr. 1.5: Pole „diferencované služby“ (DS) s vyznačeným polem DSCP [29]

Nehledě na překryv prvních tří bitů s IPP je prvních šest bitů (bity 0–5 na obr. 1.5) polem DSCP a je nedělitelné.

Dokument RFC 2597 [37] doplňuje DSCP o tzv. „zaručené doručování při přeskoku“ AF PHB (Assured Forwarding Per-Hop Behaviors) často označované pouze AF [21]. RFC 2597 [37] na str. 6 v kap. 6 definuje 4 třídy priorit, jejichž binární vyjádření je shodné s binárním vyjádřením bitů 0–2 CS a IPP. Zbývající 2 bity (bity 3 a 4 na obr. 1.5) slouží k nastavení pravděpodobnosti s jakou bude v případě potřeby paket zahozen. Budou-li na síťovém prvku 2 pakety stejné priority, bude v případě potřeby zahozen ten s vyšší pravděpodobností zahození.

Pravděpodobnost zahození je 2bitovým binárním vyjádřením dekadického čísla 1–3, přičemž 1 vyjadřuje nejnižší pravděpodobnost zahození. V praxi se udává pravděpodobnost ve tvaru AF32 (binární vyjádření `011100`), kde 32 značí, že paket patří do 3. třídy front a bude zahozen se střední pravděpodobností. Méně transparentní, i když také používané, je udávání hodnoty v dekadickém tvaru. Při převodu do dekadického tvaru se uvažuje i `0`, která je na pozici 5. b. V případě AF32 je tvar 28, což je převod binárního vyjádření do dekadického. Celá tabulka hodnot AF viz příloha A.4.

Vyšší prioritu než DSCP AF 4x má DSCP EF. Priorita EF označuje tzv. „Expedited Forwarding“, tedy česky podle [21] „urychlené doručování“. Binárně je hodnota DSCP EF vyjádřena jako `101110`. Binární vyjádření by při zápisu obvyklém u DSCP AF (kompletní výčet viz příloha A.4) odpovídalo DSCP EF 53, toto vyjádření se nepoužívá. DSCP EF nemá varianty. Z tohoto nepoužívaného vyjádření jsou však vlastnosti paketů DSCP EF nejlépe patrné. Pakety DSCP EF mají při zasílání nejvyšší prioritu, ale současně nejvyšší pravděpodobnost zahození. Použití DSCP EF je tedy možné téměř výhradně pro služby s vysokou senzitivitou na zpoždění paketů a současně s nízkou senzitivitou na jejich ztrátovost [21], [38]. Dekadicky lze DSCP EF vyjádřit číslem 46. Dokument RFC 4594 doporučuje používat DSCP EF na přenosy VoIP [39].

Ještě vyšší prioritu má CS6 (binárně `110000`) – doporučeno pro značkování směrovacích protokolů. IPP 7 (binárně `111000`), viz příloha A.3, nemá v DSCP protějšek.

Kromě DSCP AF, DSCP EF, definuje RFC 4594 ještě DSCP DF. DSCP DF (Default Forwarding) „výchozí doručování“ je výchozí hodnota. Pakety s takto nastavenou prioritou jsou zpracovávány podle metodiky „Best Effort“.

1.2.3 Hlavička IPv6 a QoS

Protokol IPv6 je definován doporučením IETF RFC 2460 [40]. To bylo vydáno v roce 1998 a později opakovaně aktualizováno. Doporučení RFC 2460 [40] nahradilo původní RFC 1883 [41] z roku 1995. Protokol IPv6, nástupce protokolu IPv4, je s IPv4 nekompatibilní. Hlavička IPv6 je od předchozí verze odlišná. Základní hlavička IPv6 má neměnnou velikost, volitelné části byly přesunuty do hlaviček rozšiřujících [42].

Z hlediska QoS obsahuje IPv6 hlavička dvě důležitá pole – pole „třída provozu" (Traffic Class) a „značka toku" (Flow Label). V RFC 2460 [40] je třída provozu definována v kap. 7.

Pole „třída provozu" je definováno jako 1B. Doporučení neuvádí konkrétní hodnoty tohoto pole. V IPv6 hlavičce je situováno od pozice 4. b (přímo následuje za 4b polem „verze protokolu"). Doporučení vyžaduje, aby bylo ve výchozím stavu pole „třída provozu" naplněno nulou [42], [40]. Pole „třída provozu" má stejné určení jako pole DS (diferencované služby) v hlavičce protokolu IPv4, viz kap. 1.2.2 a příloha A.4. Dokument RFC 2474 [29] definuje DS pro oba protokoly – IPv4 i IPv6. Hodnoty pole jsou shodné s hodnotami pole DS v IPv4.

Druhé pole důležité z hlediska QoS je pole „značka toku". Obdoba tohoto pole nebyla v IPv4 [27] definována. Značka toku je 20 b dlouhá a bezprostředně následuje za polem „třída provozu". Doporučení RFC 2460 [40] toky definuje [42], ale zpřesňující definice je dána až doporučením RFC 3697 [43]. To definuje pravidla pro práci se značkami toků [42].

Doporučení RFC 3697 [43] stanovuje příslušnost paketu k danému toku podle 3 parametrů – značky toku, IP adresy odesilatele a IP adresy cíle. Tok v IPv6 protokolu by měl usnadnit a urychlit zpracování příslušných paketů na aktivních prvcích, kterými prochází. Tokem je myšlen jeden proud přenášených dat, např. data jednoho VoIP hovoru, IPTV vysílání apod.

Toky značkuje podle RFC 3697 [43] vždy pouze odesilatel a během průchodu paketu sítí se značka nemění. Všechny pakety patřící do stejného toku mají stejnou značku. Ke značkování toku se používá celé 20b pole. Odesílající uzel přiřazuje značky toku buď pseudonáhodně, nebo postupně od 1. Pokud je v poli „tok" nula, je tak označen paket bez značky, resp. paket nepatřící do žádného toku [42].

V současné technické praxi není používání pole „tok" příliš rozšířené. Při zpracování IPv6 z hlediska QoS se v současné době využívá, obdobně jako v IPv4, prakticky výhradně pole „třída provozu".

1.2.4 Prioritní třídy protokolu MPLS

Protokol MPLS (Multi-Protocol Label Switching) je na transportních sítích hojně rozšířen. Prakticky jsou jím nahrazeny sítě Frame Relay, ATM, SDH atd. Protokol MPLS je definován doporučením RFC 3031 [44], RFC 3032 [45] a navazujícími dokumenty. Pro potřebu definice priority byly v MPLS hlavičce dedikovány 3 bity, ty následují přímo za polem „label", (značka) které je 20b. V původní definici RFC 3032 [45] je pole nazýváno „experimentální" (experimental) a značeno EXP, ačkoli bylo od počátku navrhováno jako pole pro značkování priority.

Dokument RFC 5462 [46] z roku 2009 pole EXP přejmenovává na pole „Traffic Class field", značeno „TC field", tedy na pole třídy provozu. Velikost pole a jeho další parametry zůstaly nezměněny. Chování MPLS vzhledem k DS je definováno dokumentem RFC 3270 [47], ten byl v několika ohledech aktualizován a doplněn dokumentem RFC 5129 [48].

V MPLS existují možnosti, jak řešit DS. Obvykle jsou pakety přicházející s již danou prioritou směrovány sítí pomocí MPLS značek. Cestě, po které jsou pakety přenášeny, je pomocí MPLS směrovacích protokolů nastaveny vlastnosti, které daným podmínkám vyhovují. Existuje však také možnost konvertovat informaci o prioritě do pole „TC field". Při směrování

paketu MPLS sítí P směrovače nezpracovávají informace v paketech zapouzdřených do MPLS rámce. Pokud má být na některém směrovači priorita zohledněna, musí být informace o ní udána v MPLS hlavičce.

1.2.5 *Praxe značkování na směrovačích*

Protože značkování provozu, zohledňování značek a pravidel QoS je v současné technické praxi rutinní, jsou pro značkování vyvinuty sofistikované techniky a nástroje.

Značkování probíhá ve většině případů na přístupových směrovačích sítě. Je žádoucí, aby byla značka priority do paketu (nebo rámce) přidána co možná nejblíže hraně sítě nebo zdroje provozu, viz kap. 1.2. Možnosti značkování se liší podle výrobce aktivního prvku. V rámci řešení disertační práce probíhalo testování na směrovačích Cisco 2821 a několika dalších směrovačích a přepínačích stejného výrobce, kterými je výzkumná laboratoř vybavena.

V klasifikační fázi jsou identifikovány pakety mající vybrané společné znaky náležící do požadované třídy (skupiny). Třídy jsou definovány v tzv. „mapách tříd" – skupinách znaků společných zájmovému provozu. Třídy (class) mohou být definovány mnoha způsoby, resp. pakety mohou být přiřazovány do třídy mj. podle:

- vstupního rozhraní,
- protokolu – zde obvykle aktivní prvky disponují (dle licence, výrobce a verze daného prvku) desítkami až stovkami protokolů. Mj. se může jednat o SIP, RTP, SCCP, HTTP, FTP, Skype, ICMP, IP, TCP/UDP, BGP… V rámci vybraných protokolů lze specifikaci dále zpřesňovat (např. specifikovat TCP/UDP port, IP adresu, případě podsíť apod.). Jedná se o tzv. klasifikaci MF, MultiField classification.
- VLAN,
- MPLS značek,
- délky paketu,
- DSCP, resp. CoS – umožní na aktivním prvku zohlednit dříve přidanou značku (tzv. klasifikace BA, Behavior Aggregate),
- MAC adres.

Pravidla, resp. třídy, lze řetězit. Přiřazování paketů do tříd je v konečném důsledku velmi přesné.

Jak bylo zmíněno, kromě TCP/UDP portu a IP adresy lze zpracovat i protokoly vyšších vrstev. Identifikace toků na vyšších vrstvách je náročnější a často i pomalejší oproti identifikaci na L3/L4 RM ISO/OSI. Některé aplikace ale TCP/UDP porty alokují dynamicky. Pomocí nástrojů pro klasifikaci na vyšších vrstvách lze mj. přečíst URL (Uniform Resource Locator), MIME (Multipurpose Internet Mail Extensions) hlavičky atd. Klasifikace paketů na vyšších vrstvách umožňuje např. označovat HTTP provoz směřující na specifikovanou HTTP adresu. Lze tak přiřadit různou prioritu stejnému protokolu se stejnou cílovou IP adresou, ale odlišnou doménou, nebo lze do zvolené třídy přiřadit jen VoIP provoz z určitého rozhraní a zvolené podsítě atp.

Zpracování protokolů vyšších vrstev je užitečné i v situacích, kdy v síti na jednom stejném portu komunikuje více programů, resp. protokolů. Typickým příkladem mohou být TCP porty 80 a 443 prostřednictvím kterých obvykle komunikují protokoly HTTP a HTTPS. Vzhledem k tomu, že tyto porty jsou určeny k využití nejběžnějšími internetovými službami, jsou často využívány i dalšími programy, zejména kvůli hladkému průchodu firewallem. Aktualizované databáze pro rozpoznávání protokolů vyšších vrstev jsou buď součástí firmwaru daného

prvku, nebo je možné je získat v rámci podpory výrobce samostatně (Cisco – Packet Description Language Modules, resp. technologie NBAR – Network Based Application Recognition) [21], [49].

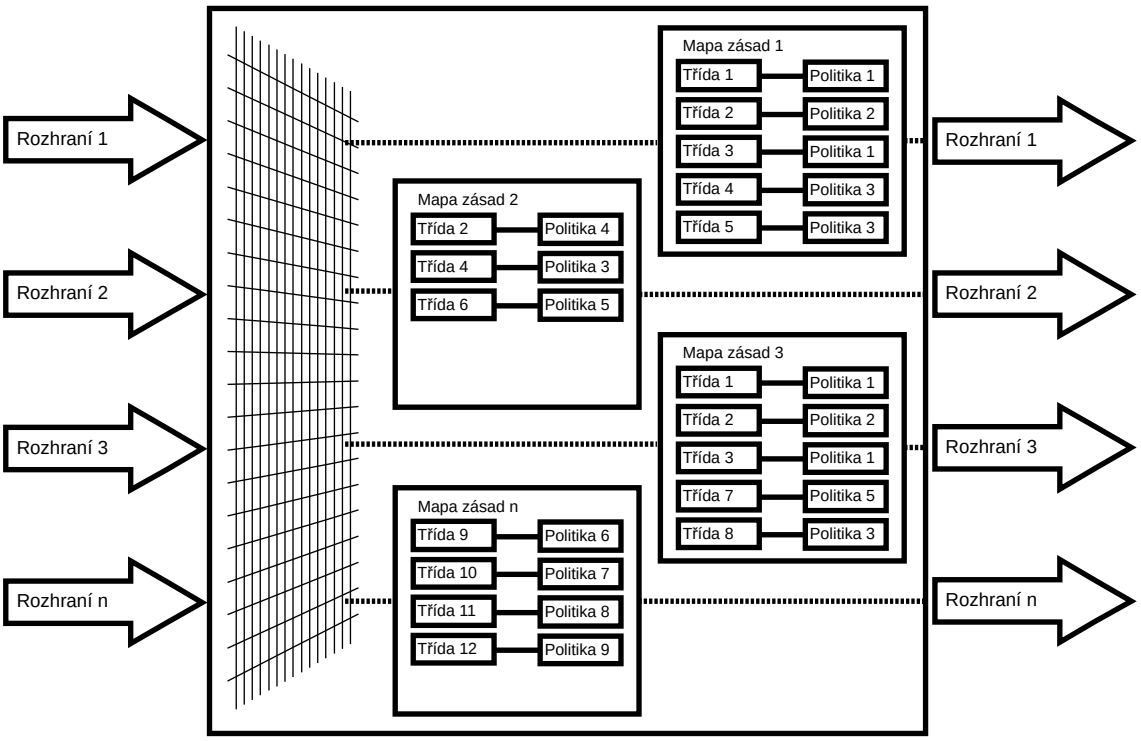
Identifikace provozu (přiřazení do třídy) šifrovaného na vyšších vrstvách je problematická, protože nelze využít rozpoznávání protokolů podle obsahu. Často je možné šifrovaný provoz zařadit do třídy pouze podle IP adres, resp. čísla TCP/UDP portu, tedy informací předávaných na 3. či 4. vrstvě RM OSI. Naopak značkování provozu šifrovaného na 2. vrstvě (IEEE 802.1ae [50]) problematické není, protože ještě před zpracováním provozu na síťovém prvku je provoz dešifrován, viz kap. 1.5.1.

Po identifikaci paketu, jeho klasifikaci – zařazení do zvolené třídy – je s paketem nakládáno podle předem definovaných zásad (politik). Zásady, jak s daným paketem pracovat, např. jak daný paket značkovat, jsou soustředěny v tzv. „mapě zásad" (policy-map). Značkuje se obvykle provoz na 3. vrstvě RM ISO/OSI, viz kap. 1.2.2 a 1.2.3. Pokud se v síti vyskytují aktivní prky, které nejsou schopny zpracovat prioritu uloženou v hlavičce 3. vrstvy, je možné informaci o prioritě přenést do hlavičky vrstvy nižší. Příkladem jsou některé (obvykle starší) síťové přepínače. Hlavička DS se převádí na CoS na aktivním prvku nejblíže k zařízení zpracovávající pouze 2. vrstvu, viz kap. 1.2.1 [21].

Kromě značkování lze provoz náležící třídě přímo omezovat (regulovat, resp. tvarovat) – upřednostňovat, přidělovat mu definovanou šířku pásma nebo naopak jej přiřadit do fronty s nižší prioritou atd. Mapa zásad náležící zvolenému rozhraní může být zpravidla jen jedna, ale v rámci 1 mapy zásad je většinou možné definovat pro každou třídu rozdílnou politiku, případně mohou být i politiky kombinovány. Paket může být v rámci politiky označkován a současně zařazen do softwarové fronty se zvolenou prioritou, viz kap. 1.3.2.

Po přiřazení paketu do třídy a aplikaci politiky na danou třídu je mapa zásad sdružující politiky nad jedním rozhraním přiřazena k rozhraní a směru provozu – je stanoveno, zda se bude mapa politik týkat vstupního nebo výstupního provozu.

Na obr. 1.6 je zachycen proces průchodu paketu/rámce čtyřportovým aktivním prvkem (typicky přepínačem) z hlediska klasifikace provozu. Dále pak schéma zachycuje aplikaci politik a jejich návaznost na třídy. Vstupní a výstupní rozhraní jsou oddělena. Vlevo jsou zachycena vstupní rozhrání (provoz), vpravo výstupní.



Obr. 1.6: Průchod paketu aktivním prvkem

1.2.6 Doporučené hodnoty CoS a DSCP

Pro nastavení hodnot polí CoS, DSCP a jejich generačně starších variant, zmíněných v kapitolách 1.2.1, 1.2.2 a 1.2.3, jsou stanovena obecně dodržovaná doporučení daná IEEE 802.1Q [51] pro CoS a dokumenty RFC 4594 [39] a RFC 5865 [52] pro QoS.

Příkladem využití upřednostňování (prioritizace) provozu může být např. značkování protokolů VoIP, IPTV, videokonferenčních přenosů atd. Nastavení příznaků pole DSCP tokům v síti je závislé na konkrétních podmínkách dané konkrétní sítě a zejména požadovaným výsledkům. Většinou jsou ale typické toky, jakými je mj. zmíněný VoIP, značkovány dle doporučení. Je-li v síti používáno DSCP, je doporučeno značkovat VoIP provoz prioritou DSCP EF, videokonference DSCP AF41. Opačnou částí jsou velkoobjemové datové přenosy s doporučením DSCP AF11.

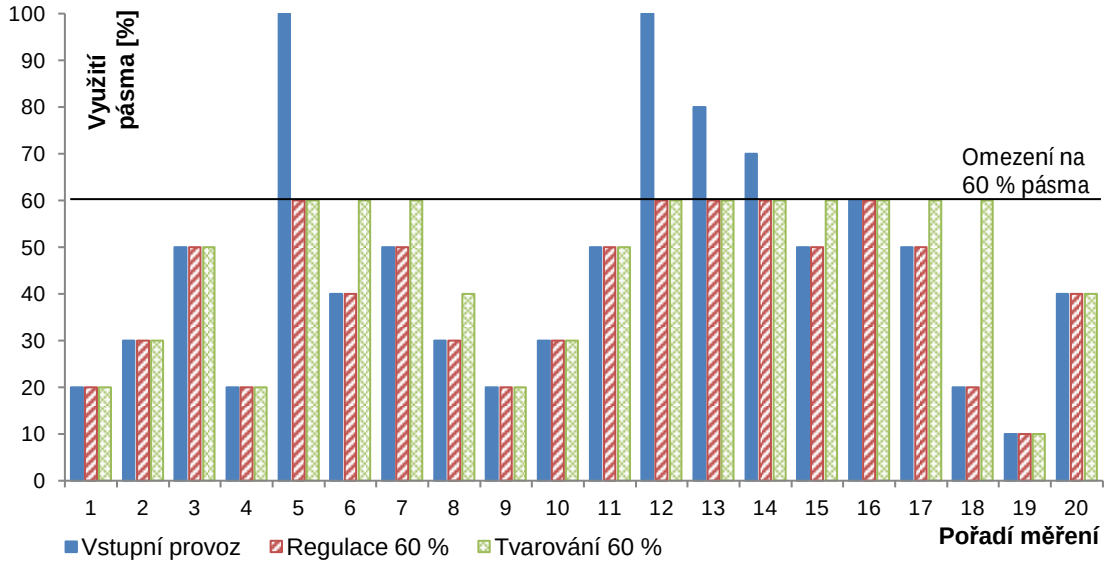
Pokud se v síti nepoužívá DSCP AF, je doporučeno značkovat samotný provoz VoIP (protokol RTP) značkou CS5, signalizaci pak značkou CS3. Dále např. videopřenosy (streamované video) značkou CS4 atd., viz [39], [51] a [52]. Pokud pole DSCP nebo CoS je vyplněno nulami, je s paketem nakládáno podle politiky „Best Effort“.

1.2.7 Regulace a tvarování provozu – omezování šířky pásma

Kromě nastavení priority vybranému paketu je v aktivním prvku třeba prioritu vyhodnotit a s daným paketem podle ní naložit. Součástí politik popsaných v kap. 1.2.5 jsou i možnosti, resp. algoritmy, pomocí nichž lze omezit spotřebovávanou šířku pásma definovaného toku. Jedná se o 3 algoritmy – nejpoužívanější algoritmus „Token Bucket“, následně pak „Leaky Bucket“ a „Virtual scheduling algorithm“ [6].

Omezit šířku pásma lze dvěma hlavními postupy – zahozením (případně úpravou priority) paketů, které ve zvoleném toku stanovenou šířku pásma přesahují, tzv. regulace (policing), nebo lze tyto pakety uložit do fronty (opět s možností úpravy jejich priority) a odeslat později, tzv. formování označované také jako tvarování (shaping) [21].

Příklad omezování šířky pásma je zachycen na obr. 1.7, na kterém je omezován datový tok na 60 %. Provoz lze regulovat zpravidla jak předepsáním povoleného procentuálního využití šířky pásma rozhraní, tak pevně stanovenou přenosovou rychlostí. Modré sloupce na obr. 1.7 symbolizují provoz na vstupním rozhraní aktivního prvku. Červenou barvou je vyznačen výstupní provoz po použití „regulačních“ algoritmů, zeleně je vyznačen „tvarovaný“ provoz. Z obr. 1.7 je patrné, že regulovaný provoz kopíruje provoz vstupní, vyjma okamžiků, kdy překračuje stanovenou šířku pásma. Zelený, tvarovaný, provoz využívá softwarových front, viz kap. 1.3.2 a provoz přesahující omezující hranici do nich ukládá. Provoz na výstupu je pak rovnoměrnější. To je na obr. 1.7 pozorovatelné zejména v časech (taktech) 5–8 a 12–18. Fronty použité při tvarování mají omezenou velikost – jednak z kapacitních důvodů, ale také kvůli vnesenému zpoždění, viz dále kap. 1.3.2. Chování obou mechanismů je závislé na nastavení jejich parametrů, tedy i graf na obr. 1.7 by měl se změnou parametrů odlišný průběh. Cílem grafu je znázornit obecný princip, místo konkrétního příkladu.



Obr. 1.7: Ilustrace omezování šířky pásma

Regulace provozu je nejčastěji realizována algoritmem „Token Bucket“, případně rozšířenou variantou „Dual Token Bucket“. Algoritmus „Token Bucket“ (bucket – v češtině kbelík, vědro; dle [21] překládáno jako „tokenový zásobník“) odešle paket pouze tehdy, má-li k dispozici token. Token přitom není vázán na paket, ale na bit (regulace je nezávislá na velikosti právě zpracovávaného paketu). Tokeny jsou přidělovány rychlostí odpovídající požadované výstupní šířce pásma regulovaného provozu. „Token Bucket“ je algoritmus označován proto, že lze průměrnou přenosovou rychlost krátkodobě překročit v závislosti na počtu tokenů uložených v tokenovém zásobníku. Rychlost, s jakou jsou tokeny přidávány do tokenového zásobníku, je stanovena rovnicí (1.3) [21], kde:

- c_T označuje počet tokenů přidaných do tokenového zásobníku,
- t_{p_n} [s] označuje čas příchodu n. paketu, resp. $t_{p_n} - t_{p_{n-1}}$ dobu (rozdíl času) mezi příchodem dvou po sobě jdoucích paketů.
- CIR [b/s] – „Committed Information Rate“, potvrzená přenosová rychlost. Obvykle je rovna tzv. „tvarové rychlosti“ nebo „regulační rychlosti“, tedy rychlosti, na kterou se bude provoz omezovat (tvarovat nebo regulovat).

$$c_T = \left(t_{p_n} - t_{p_{n-1}}\right) \cdot CIR \tag{1.3}$$

Podle typu a výrobce aktivního prvku lze celou pravou stranu rovnice (1.3) podělit konstantou 8 – to platí mj. pro aktivní prvky Cisco, kde je token vázán na bajt.

Tokenový zásobník má velikost B_c, pokud jsou podporovány „nadměrné shluky" (Excess Burst). Pak je velikost tokenového zásobníku rovna $B_c + B_e$.

- B_c (b) – „Committed Burst Size" (dle [21] česky „potvrzená velikost shluku") určuje, kolik bitů je možné odeslat v rámci 1 periody, tedy za dobu T_c. Potvrzená velikost shluku je v některých aktivních prvcích udávána v bajtech. Potvrzené velikosti shluku odpovídá rovněž velikost tokenového zásobníku.
- B_e (b) – „Excess Burst Size" (dle [21] „velikost nadměrného shluku") stanovuje, kolik bitů „tokenů" lze nad rámec B_c uložit do tokenového zásobníku.
- T_c [s] – označuje časový interval, po který je možné odesílat shluk B_c. Časový interval T_c je obvykle dopočítáván aktivním prvkem podle vztahu (1.4) [21].
- PIR (b/s) – „Peak Information Rate", špičková přenosová rychlost [21].

V praxi, při konfiguraci různých aktivních prvků různých výrobců, ale i modelových řád a firmwarů stejného výrobce, je často třeba opatrnosti – parametry nutné pro nastavování tvarování a regulace udávají poměrně nesystematicky v některých případech v bitech, v jiných v bajtech.

$$T_c = \frac{B_c}{CIR} \tag{1.4}$$

Regulovat lze jak vstupní, tak výstupní provoz na rozhraní. Regulace algoritmem „token bucket" rozlišuje 3 typy paketů, přičemž poslední uvedený je využíván pouze v rozšířené variantě algoritmu „token bucket", tzv. „dual token bucket", viz dále. Rozlišovány jsou pakety:

- „vyhovující (conforming)", jejichž $v_p \in (0; CIR\rangle$, tedy pakety, které přichází rychlostí menší než je CIR,
- „překračující (exceeding)", jejichž $v_p \in (CIR; \infty\rangle$ pro algoritmus „token bucket" a $v_p \in (CIR; PIR\rangle$ pro algoritmus „dual token bucket",
- „narušující (violating)", jejichž $v_p \in (PIR; \infty)$, pouze pro „dual token bucket".

Vyhovující pakety nejsou regulovány a jsou odesílány k dalšímu zpracování. Pokud je použit nejjednodušší algoritmus „token bucket", jsou překračující pakety, v závislosti na nastavené politice – viz kap. 1.2.5 – buď zahozeny, nebo např. přeznačkovány tak, že je zvýšena pravděpodobnost jejich zahození.

Sofistikovanější nakládání s překračujícími pakety umožňuje algoritmus „dual token bucket", který umožňuje zavedení další kategorie paketu, tzv. pakety „narušující". Algoritmus „dual token bucket" přidává 2. tokenový zásobník o velikosti B_e, do kterého jsou ukládány tokeny, které svým množstvím překračují velikost prvního tokenového zásobníku velikosti B_e. Algoritmus naplňování obou zásobníků novým tokenem je popsán výroky (1.5), kde $c_{T_{B_c}}$ vyjadřuje počet tokenů v prvním tokenovém zásobníku velikosti B_c a $c_{T_{B_e}}$ počet tokenů ve druhém tokenovém zásobníku velikosti B_e. Rychlost naplňování zásobníků je i v tomto případě definována rovnicí (1.3).

$$
\begin{aligned}
c_{T_{B_{c_n}}} = B_c \wedge c_{T_{B_{e_n}}} = B_e &\Rightarrow c_{T_{B_{c_{n+1}}}} = B_{c_n} \wedge c_{T_{B_{e_{n+1}}}} = B_e \\
c_{T_{B_{c_n}}} = B_c \wedge c_{T_{B_{e_n}}} < B_e &\Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} + 1 \\
c_{T_{B_{c_n}}} < B_c &\Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} + 1 \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}}
\end{aligned}
\tag{1.5}
$$

Podoba algoritmu „dual token bucket“ popsaná výroky (1.5) umožňuje regulovat provoz na konkrétní rychlost a současně umožňuje propouštět nadměrné shluky. Vyprazdňování zásobníků v případě „vyhovujícího“ paketu je popsáno výrokem (1.6) [21]. c_{T_o} označuje počet tokenů potřebných pro odeslání daného paketu – tento počet tokenů je třeba odebrat z tokenových zásobníků.

$$
c_{T_o} \leq c_{T_{B_{c_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} - c_{T_o} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} \tag{1.6}
$$

Splní-li paket podmínky stanovené výrokem (1.7), půjde o paket „překračující“ a tokeny budou odebrány z druhého tokenového zásobníku velikosti B_e [21].

$$
c_{T_o} > c_{T_{B_{c_n}}} \wedge c_{T_o} \leq c_{T_{B_{e_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} - c_{T_o} \tag{1.7}
$$

Nesplní-li paket ani podmínky stanovené vztahem (1.6) ani výrokem (1.7), jedná se o paket „narušující“. Tento algoritmus je popsán vztahem (1.8) [21].

$$
c_{T_o} > c_{T_{B_{c_n}}} \wedge c_{T_o} > c_{T_{B_{e_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} \tag{1.8}
$$

Popsaný algoritmus „dual token bucket“ má několik variant. Varianta popsaná výroky (1.5)–(1.8) je tzv. „Single-Rate“ (jednorychlostní). Varianta, která umožňuje nadměrné shluky odesílat trvale, se nazývá „Two-Rate“ (dvourychlostní). Tokenové zásobníky velikosti B_c a B_e se v této variantě algoritmu naplňují tokeny samostatně, nezávisle, dvěma různými rychlostmi c_T odvozenými od CIR a PIR. Obě rychlosti naplňování zásobníků jsou vypočteny podle stejného vztahu (1.3), ale v případě $c_{T_{PIR}}$ s dosazením PIR namísto CIR. Obdobně jako v případě jednorychlostní varianty „dual token bucket“ algoritmu jsou tokeny přidané do obou tokenových zásobníků nad rámec jejich velikosti zahozeny.

Odesílání paketů, resp. vyprazdňování obou tokenových zásobníků dvourychlostního algoritmu „dual token bucket“, je definováno výrokem (1.9) pro případ „vyhovujících“ paketů.

$$
c_{T_o} \leq c_{T_{B_{c_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} - c_{T_o} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} - c_{T_o} \tag{1.9}
$$

Překračující provoz a s tím související vyprazdňování tokenových zásobníků, je definováno výrokem (1.10) [21].

$$
c_{T_o} > c_{T_{B_{c_n}}} \wedge c_{T_o} \leq c_{T_{B_{e_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} - c_{T_o} \tag{1.10}
$$

Pro narušující provoz a související vyprazdňování zásobníků platí výrok (1.11) [21].

$$
c_{T_o} > c_{T_{B_{c_n}}} \wedge c_{T_o} > c_{T_{B_{e_n}}} \Rightarrow c_{T_{B_{c_{n+1}}}} = c_{T_{B_{c_n}}} \wedge c_{T_{B_{e_{n+1}}}} = c_{T_{B_{e_n}}} \tag{1.11}
$$

Tvarování provozu je druhý z mechanismů omezování šířky pásma. Tvarování provozu, na rozdíl od regulace, nezahazuje všechny pakety, které překračují stanovenou hranici. Pakety, jsou ukládány před odesláním do tzv. „tvarovací fronty“ (shaping queue). Tvarovací fronta je

v základní podobě tvarování typu FIFO, viz kap. 1.3. Při tvarování se používá nejčastěji algoritmus „Token Bucket“ popsaný výše v této kapitole. Rozhraní odesílá pakety linkovou rychlostí. Tvarovač ale plánuje odesílání paketů tak, aby byla průměrná rychlost rovna rychlosti tvarové. Toto je dosaženo odesíláním paketů v periodických intervalech střídaných „tichem“. Doba, po kterou je povoleno odsílání T_c, je definována rovnicí (1.4). Velikost shluku B_c je definována konfigurací aktivního prvku.

Cisco podle [21] doporučuje při tvarových rychlostech do 320 kb/s stanovit konstantně B_c = 8 000 b a T_c vypočítat dle rovnice (1.4). Při tvarových rychlostech vyšších než 320 kb/s naopak ponechat konstantně nastavenu velikost T_c = 25 ms a B_c , vyjádřit a vypočítat ze vztahu (1.4). Dále je dle [21] doporučeno ponechat velikost nadměrného shluku B_e = B_c. Pokud není v tokenovém zásobníku dostatek tokenů pro odeslání potřebného množství bitů (paketů), jsou pakety uchovávány ve tvarovací frontě. Kvůli zachování efektivity je ale velikost tvarovací fronty omezená. Jestliže je v tokenovém zásobníku nedostatek tokenů trvalý, jsou pakety přicházející do tvarovací fronty zahozeny.

Tvarování provozu je oblast provázaná s problematikou softwarových front, viz kap. 1.3.2 a 1.3.3. Pokud je např. aktivní LLQ (Low Latency Queuing), zařazují se pakety namísto jedné tvarovací fronty typu FIFO do více front v závislosti na klasifikaci – typu (třídě) – provozu. Z prioritních front jsou pak pakety odesílány tvarovačem přednostně před pakety uloženými ve frontě s nižší prioritou. Tvarovač odesílá paket na rozhraní, kde je (podle konfigurace) buď zařazen do hardwarové fronty a následně odeslán, nebo je podle třídy paket zařazen do softwarové fronty – dále viz kap. 1.3.2.

Tvarování se na rozhraní aktivuje v okamžiku, kdy je překročena rychlost CIR. Pokud se vyprázdní tvarovací fronty, rychlost vstupujících poklesne pod CIR, tvarování se nepoužívá a pakety jsou odesílány přímo do softwarových/hardwarových front náležícím jednotlivým výstupním rozhraním.

Pozn. Regulaci a tvarování lze využít i jako podporu při ochraně před DoS, resp. i DDoS (Distributed Denial of Service, distribuované odepření služby), de facto se jedná o jeden z primárních účelů těchto mechanismů. Pomocí omezování šířky pásma některých služeb, uživatelů, uzlů,… je možné zajistit klíčovým službám, resp. jejich uživatelům, dostatečnou šířku pásma pro užívání těchto „klíčových služeb“. Protože ne každý DoS musí být nutně kybernetickým útokem (příkladem takového DoS je přetížení zpravodajských serverů v průběhu mimořádných událostí), nemusí být úplné zahození všech „problematických“ paketů nejlepším řešením. Pomocí NAC (Network Access Control) a dalších mechanismů lze zkontrolovat systém uživatele s vyšší prioritou přístupu a adekvátně automatizovaně nastavit aktivní prvky. Pomocí regulace, tvarování a prioritizace provozu je možné „prioritním“ uživatelům a službám zajistit dostatečnou šířku pásma. V této souvislosti a v souvislosti s SDN (Software-Defined Networking, softwarově definované sítě) se často hovoří o tzv. „Follow Me QoS“, tedy svázání pravidel QoS s konkrétním uživatelem a jím provozovanými (využívanými) síťovým službami. Taková QoS pravidla mohou být pro uživatele neměnná bez ohledu na jeho mobilitu v síti, nebo naopak mohou na uživatelovu mobilitu pružně reagovat.

1.3 Fronty a jejich zpracování

Fronty (anglicky queues) jsou konstrukční částí aktivních síťových prvků. Správa a řízení front je jedním z mechanismů, kterými lze ovlivňovat pořadí a rychlost, jakou jsou rámce/pakety

v síti doručovány. Fronty jsou jednou z klíčových částí systému zajišťující v aktivním prvku QoS. Fronty slouží k uložení rámce před jeho zpracováním nebo odesláním.

Značkování paketů je v aktivních prvcích funkcionalita oddělená od nakládání s označeným provozem. O problematice značkování a zpracování polí QoS pojednává kap. 1.2 a zejména pak kap. 1.2.5.

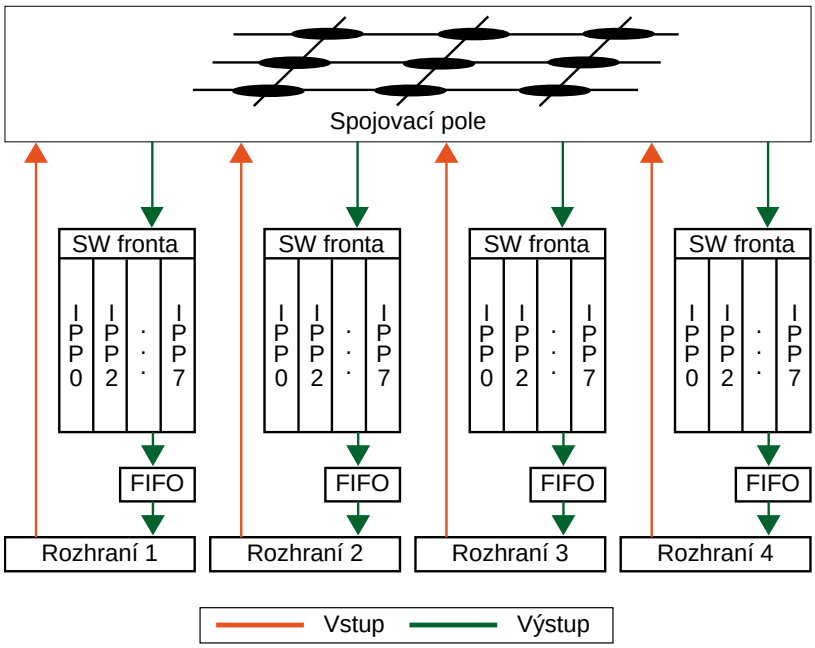
Technicky je nakládání s prioritami zapsanými v paketech (viz kap. 1.2.1 až 1.2.4) realizováno mj. pomocí správy front (queue management). V síťových prvcích lze identifikovat 2 klíčové typy front – softwarové a hardwarové. Paket se ve frontách zdržuje pouze tehdy, je-li vstupní datový tok v daném okamžiku větší než výstupní, resp. není-li přijímací prvek připraven přijímat provoz. To platí jak pro datové toky vstupující do aktivního prvku, tak pro toky vstupující do určitých částí uvnitř aktivního prvku, viz dále. Fronty svojí existencí umožňují zpracování paketů, které by byly jinak zahozeny, což celkově zvyšuje efektivitu přenosů na síti (viz kap. 1.1). Je-li situace opačná – je-li k dispozici dostatečná šířka pásma na výstupu aktivního prvku nebo je-li daná část schopna okamžitě paket zpracovat, paket se do softwarové, resp. ani hardwarové fronty nezařazuje a přímo se odesílá na výstupní rozhraní. Při jakémkoli upřednostňování vybraných toků se současně jiné toky nevyhnutelně zpožďují. Upřednostňování je na jejich úkor.

Pakety se do softwarových front zařazují až v okamžiku, kdy je na hardwarových frontách indikován tzv. „stav zahlcení“ – tedy když je hardwarová fronta plná [21]. Standardně se pakety nejprve odesílají přímo na výstupní rozhraní. Pokud není schopna protistrana pakety přijímat, ukládají se do hardwarových FIFO front přímo na výstupech rozhraní. Pokud jsou hardwarové fronty plné, začnou se pakety ukládat a zpracovávat ve frontách softwarových.

Při zaplnění softwarových front se uplatňují mechanismy zahazování, viz kap. 1.3.4. Pokud je to nutné, paket se nejprve zpracuje v softwarové frontě a teprve v případě, že jej nelze přímo odeslat, se uloží do hardwarové fronty. Softwarové (ani hardwarové) fronty nebývají používány na čistě softwarových rozhraních. Zpracování front je v síťových prvcích různých výrobců různé.

Nastavení velikostí hardwarových a softwarových front je svázáno s problematikou QoS. Obecně je velikost front kompromisem mezi velikostí vneseného zpoždění a schopností absorbovat shluky (bursty) paketů [53], viz dále.

Schematicky je příklad uspořádání softwarových a hardwarových front v aktivním prvku zachycen na obr. 1.8. Znázorněn je model aktivního prvku se čtyřmi vstupně-výstupními rozhraními (typicky čtyřportový přepínač). Každému výstupnímu rozhraní na obr. 1.8 náleží jedna hardwarová fronta a osm front softwarových odpovídající osmi prioritám IPP, viz kap. 1.2.2. V praxi ale není použití tak vysokého počtu softwarových front nad jedním rozhraním obvyklé. Pro přehlednost obr. 1.8 nezachycuje přijímací fronty na vstupních rozhraních.



Obr. 1.8: Schématické znázornění použití front v aktivním prvku

1.3.1 *HARDWAROVÉ FRONTY*

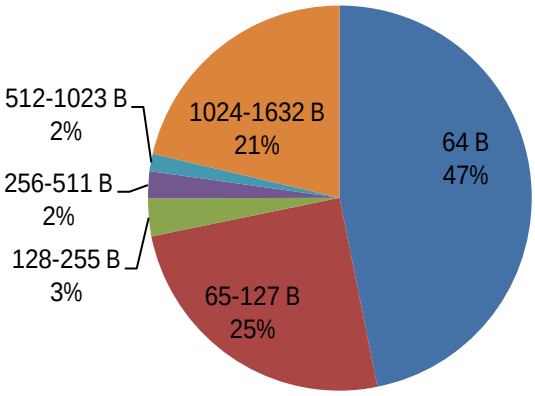
Hardwarové fronty jsou výlučně typu FIFO (First In, First Out, česky první dovnitř, první ven) – zachovávají pořadí paketů vstupujících a vystupujících z fronty [21]. FIFO fronty slouží v aktivních prvcích jako vyrovnávací paměť na rozhraních. Hardwarové fronty jsou vždy součástí každého fyzického vstupně-výstupního rozhraní, přičemž jednomu rozhraní obvykle náleží dvě fronty – přijímací (receive) a vysílací (transmit). Na zařízeních Cisco jsou fronty označovány, podle typu zařízení, jako „vysílací/přijímací fronta“ (transmit/receive queue) nebo „vysílací/přijímací okruh“ (transmit/receive ring) [21].

V hardwarové frontě je uložen celý rámec (včetně preambule) do okamžiku, než je protilehlé rozhraní připraveno jej přijmout. Čtení hardwarových front není řízeno CPU (Central Processing Unit – procesor) aktivního prvku, což šetří procesorový čas a zvyšuje výkon, resp. propustnost daného zařízení.

Hardwarové fronty nejsou softwarově konfigurovatelné, vyjma jejich velikosti. Ta je mimoto ovlivněna také velikostí paměti osazené v daném aktivním prvku. Při aktivaci softwarových front se velikost výstupních hardwarových front cíleně zmenšuje (viz kap. 1.3.2). Hlavním důvodem pro zmenšování jejich velikosti je zmenšování zpoždění, které do přenosu vnáší, tedy zejména snaha uložit přijatelné maximum rámců do front softwarových, kde lze s uloženými rámci pracovat.

Nastavení optimální velikosti výstupních hardwarových front je závislé na přenosové rychlosti rozhraní, ale také dalších faktorech. Velikost by podle [53] měla být nastavena tak, aby bylo možné do fronty uložit 4 rámce. Přitom je třeba uvažovat rámce dosahující velikosti MTU (Maximum Transmission Unit) v dané síti [3], [53]. Velikost front musí tedy být nastavena vždy podle konkrétních provozních podmínek dané sítě. Odchýlení velikosti fronty oběma směry od doporučené, bez důvodu, má negativní dopad na výkon a propustnost sítě. Je-li hardwarová fronta příliš velká, zbytečně může v síti narůstat zpoždění, ale i jitter, protože jsou doručovány rámce, které by za jiných okolností byly zahozeny. Za jiných okolností ale fronty pomáhají jitter eliminovat. Součástí některých aktivních prvků jsou k tomu dedikované fronty, tzv. de-jitter buffer.

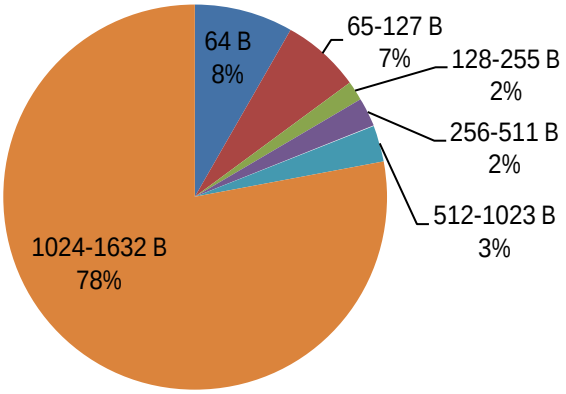
Jedním z měření, která byla v rámci vlastního výzkumu provedena, bylo měření zaměřené na sběr statistických údajů o síťovém provozu. Příklad rozložení velikostí rámců v síti 12 počítačů a 4 IP telefonů připojených k Internetu je zachycen na obr. 1.9. Měření bylo provedeno analytickými nástroji přepínače Cisco SGE 2010P a analyzátorem VeEX VePAL TX300/e na 1000Base-T Ethernetu. Přepínač tvořil střed hvězdicové topologie testované sítě. Měření bylo provedeno na propojovacím (trunk) portu, kterým byla síť připojena do Internetu. Měření probíhalo po dobu 24 dnů a 16 hodin kancelářské práce – prohlížení webových stránek, stahování PDF souborů atd. Protokol HTTP tvořil 83 % provozu, ostatní protokoly (RTP, RDP, FTP, DHCP,…) spotřebovaly zbývajících 17 %.



Obr. 1.9: Příklad rozložení velikostí rámců v síti

Unikátnost změřeného provozu, resp. zásadní odlišnost oproti „obvyklému“ internetovému provozu, podtrhuje potřebu obezřetnosti při konfiguraci hardwarových front. Během měření byla, po přepočtení na přenesená užitečná data, transportována velkými rámci velikosti 1 024–1 632 B většina (78 %) obsahu. Z toho neměřená část paketů přesahuje velikost 1 518 B (maximální velikost ethernetového rámce bez preambule a oddělovače). Jedná se o tzv. „jumbo“ rámce. V rámci internetového provozu jsou však obvykle nejpoužívanější rámce velikosti 512 B, které tvoří přibližně 90 % celkového provozu [53]. Jejich podíl na celkovém přenosu byl v testovacím případě zanedbatelný, viz obr. 1.10.

Pakety velikosti 64 B obvykle přenášejí informace o signalizaci, případně jsou používány protokoly jako je SSH, Telnet apod. K přenosům souborů se používají velké pakety, obvykle 1 500B. Pakety velikosti 128–512 B jsou v obecném internetovém provozu nejobvyklejší a pokrývají nejširší spektrum služeb. I v tomto případě se výsledky měření rozcházejí s rozložením udávaným v [53]. Rozdílnost výsledků nedokazuje chybnost závěrů uvedených v [53], ale poukazuje na to, že zásadní význam na charakteristiku provozu má lokalita, ve které je prováděno měření a to zejména s postupem směrem k nižším úrovním v hierarchii sítě.



Obr. 1.10: Rozložení velikostí rámců po přepočtení na přenesená data

Nad hardwarovými frontami nejsou prováděny žádné operace QoS. Proto příliš velká hardwarová fronta může snížit účinnost nastavených pravidel QoS realizovaných softwarovými frontami, viz kap. 1.3.2. Malá hardwarová fronta může být zase příčinou nárůstu ztrátovosti, protože v případě přetečení její kapacity je paket zahozen.

Stanovení velikosti hardwarové fronty je závislé zejména na parametrech sítě, do které je aktivní prvek nasazován. Ovlivňujícími faktory jsou:

Velikost paketů – je třeba identifikovat převládající a také maximální a minimální velikost paketů v dané síti. Přepínače pro správu paketů v hardwarových frontách používají tzv. „particle“ (česky částice, částečky) někdy označované také jako buňky. „Particle“ jsou dále nedělitelné. Jejich velikost se stanovuje podle převládající velikosti paketů v síti. Je-li rámec větší než jedna buňka, je uložen (rozdělen) do více těchto buněk (rozdělení na „particle“ je fragmentací uvnitř aktivního prvku). Z výkonnostního hlediska je výhodné, pokud velikost buněk přesně odpovídá velikosti rámců. Rámce pak není nutné fragmentovat, ale ani se neplýtvá kapacitou, jako pokud by rámce byly menší. V aktivních prvcích Cisco je velikost „particle“ stanovena na 576 B, což by mělo umožnit uložení 90 % v Internetu používaných rámců bez jejich dělení [53]. Pokud jsou pakety větší než je velikost „particle“, jsou rozděleny do více „particle“. „Particle“ je dále nedělitelná a může obsahovat pouze 1 paket. Po fyzické stránce jsou „particle“ realizovány nejčastěji pamětí SDRAM (synchronous dynamic random access memory). Např. v ATM adaptéru PA-A3 pro směrovače Cisco 7200 je tato paměť velká 4 MB [54].

Shluky (burst) – alokovaná velikost paměti současně určuje maximální velikost shluku paketů, označovanou MBS (Maximum Burst Size), kterou je aktivní prvek schopen absorbovat.

Povolené zpoždění – služby provozované na síti stanovují maximální zpoždění, jaké mohou pakety v síti nabrat, viz kap. 1.1. Pozitivním důsledkem zmenšování hardwarových front je redukce vnášeného zpoždění.

Rychlost přenosové linky – aby byla zachována účinnost hardwarových front, musí se jejich velikost odvíjet od maximální přenosové rychlosti dané linky. Větší datový tok znamená případnou potřebu uložit více dat za stejný čas.

Na aktivních prvcích se velikost hardwarových front nastavuje individuálně pro každý z portů, tzn. možnost přizpůsobení konkrétní síti.

Z pohledu QoS obsahuje hardwarová FIFO fronta na daném rozhraní pouze jednu třídu provozu. Snahou „správce front“ je proto do hardwarových front pakety nezařazovat, pro-

tože je v nich nelze kontrolovat. To se řeší minimalizací velikosti hardwarových front. S tím souvisí požadavek pakety, pokud možno, přímo odesílat. V případě, že se nachází data v softwarových frontách, je hardwarová fronta zcela zaplněna. V případě prázdné hardwarové fronty se pakety do fronty softwarové obvykle vůbec nezařazují.

V prvcích, ve kterých nejsou aktivní softwarové fronty, se v případě přetečení hardwarové fronty pakety přímo zahazují – vždy se zahazují pakety poslední příchozí. Tento mechanismus je v angličtině nazýván „tail drop“, viz kap. 1.3.4 [55]. Pakety připravené ve FIFO frontě k odeslání se nezahazují.

1.3.2 SOFTWAROVÉ FRONTY

Uplatňování pravidel QoS probíhá v tzv. „softwarových frontách“. Ty jsou implementovány a řízeny softwarově. Softwarové fronty nejsou implementovány bezvýhradně do všech aktivních prvků všech výrobců, zejména prvky určené pro segment SOHO (Small Office/Home Office, tedy malé a domácí kanceláře) a prvky bez možnosti uživatelské konfigurace (managementu) nebývají níže popsanými technologiemi a mechanismy vybavovány. Řízení softwarových front může být, ve srovnání s hardwarovými frontami, sofistikované a poskytuje rozsáhlé možnosti konfigurace. Obsluha softwarových front, v závislosti na implementaci, spotřebovává procesorový čas CPU. Délka softwarových front se obvykle stanovuje na maximálně cca 20 paketů, jedná se také o výchozí počet paketů v softwarových frontách u Cisco prvků [56].

Výrobci aktivních prvků (mj. Cisco, Juniper a Brocade, viz např. [57], [58] a [59]) implementují ve svých prvcích všechny níže uvedené metody správy front i mechanismy zahazování (Congestion Avoidance, ochrana před zahlcením). Ne všechny metody jsou ale implementovány ve všech typech a výrobních řadách aktivních prvků, které vyjmenovaní producenti dodávají. Implementace jednotlivých metod je závislá na předpokládané roli daného prvku a s tím související kategorií použití vybraného zařízení. Z technického pohledu se jedná především o velikosti pamětí, vybaveností ASIC (Application-Specific Integrated Circuit) procesory atd. I v rámci jednoho šasi a se stejným supervizorem záleží dostupnost mechanismů správy front na použitých linkových kartách, resp. kartách MPC (Modular Port Concentrator) v případě Juniperu, které jsou vybaveny příslušnými paměťmi a dalším HW určeným pro plnění úkolů spojených se správou softwarových i hardwarových front. Různým způsobem jsou omezovány také kapacity a počty softwarových front náležící rozhraní. Např. Cisco Catalyst 2960-XR, stejně jako Cisco Catalyst 3850 s IOS XE, umožňuje na jednom rozhraní (portu) použití 8 front. Na jiných zařízeních, např. Cisco Catalyst 6800 v kombinaci s technologií Instant Access a klientem („vzdálenou“ linkovou kartou) Cisco 6848ia, jsou pevně definované fronty a k nim pevně přiřazené prioritní třídy CoS/DSCP (v tomto případě se předpokládá, že omezení bude v budoucnu odstraněno)[60].

Zásady QoS, resp. softwarové fronty obecně, se uplatňují jen tehdy, jsou-li hardwarové fronty zcela zaplněny. Při aktivaci softwarových front jsou hardwarové fronty firmwarem aktivního prvku maximálně zkráceny, aby při uložení stejného množství paketů do front byla jejich podstatná část uložena v softwarových frontách. Důvody pro zkrácení hardwarových front plynou z jejich FIFO povahy. Nelze v nich, na rozdíl od SW front, uplatnit pokročilou správu a tedy zajišťovat QoS. Rámce ze všech softwarových front směřující na dané rozhraní jsou při odeslání zařazovány do jediné hardwarové (FIFO) fronty, viz kap. 1.3.1

Odesílání (vyřazování) paketů ze softwarové fronty zajišťuje plánovač paketů (Packet Scheduler), který je označován také zkráceně jako plánovač (scheduler) [55]. Úlohou plánovače je plánování, tedy stanovení, který paket bude z fronty v dalším kroku vyřazen – odeslán do

výstupní hardwarové fronty (na síťové rozhraní). Rozlišuje se plánování v rámci jedné fronty a v rámci všech front [21]. Plánovač je obvykle částí síťového procesoru NPU (Network Processor Unit) – příklady jeho možného návrhu, viz [61], popřípadě [62].

Pokud zaplnění softwarové fronty dosahuje stanovené míry, jsou pakety zahazovány podobně jako v případě front hardwarových. Při zahazování paketů v softwarové frontě je možné kromě nejjednoduššího mechanismu „tail drop" použít i několik pokročilejších mechanismů zahazování. Zahazování paketů, resp. správa obsahu front (zásobníků) je úkolem správce front (Queue Manager), ten je označován také jako správce zásobníků (Buffer Manager), viz kap. 1.3.4.

Řešení správy front a použité algoritmy jsou dány výrobcem konkrétního aktivního prvku. Softwarové fronty jsou obvykle uspořádány podle rozhraní, jímž daná fronta náleží a dále podle toků, resp. podle zařazení do tříd v klasifikační fázi, viz kap. 1.2.5. Softwarové frontě, resp. toku, třídě je v aktivním prvku konfigurací přidělena šířka přenosového pásma. Tato část šířky pásma je částí celkové dostupné šířky pásma odesílacího rozhraní. Pokud je některá z front prázdná, je v obvyklých konfiguracích šířka pásma přidělená této prázdné frontě, tedy rozdělena mezi zbývající fronty.

Přepínače a směrovače, které mají podporu softwarových front, ji mohou mít v tovární konfiguraci neaktivní. Aby bylo nasazení softwarových front efektivní, je třeba je zkonfigurovat. Jedinou výjimkou jsou softwarové FIFO fronty.

1.3.3 Mechanismy správy softwarových front

Implementace a použití softwarových FIFO front je ze všech níže popsaných metod správy softwarových front nejjednodušší, bývají proto ve většině aktivních prvků nastaveny jako výchozí, i když jejich použití není ve všech situacích nejvhodnější (může být příčinou jitteru). Softwarová verze FIFO front je funkční obdobou FIFO front hardwarových (viz kap. 1.3.1). Neumožňují tedy upřednostňovat vybrané typy provozu na úkor jiných. Podobně jako v případě hardwarových FIFO front je zahazování paketů řešeno mechanismem „Tail Drop", viz kap. 1.3.4.

V zařízeních Cisco, ale i ostatních výrobců, se kromě FIFO používá několik dalších mechanismů správy softwarových front. Mezi původní, nyní již zastaralé, algoritmy patří: WFQ (Weighted Fair Queuing, vážené fronty), PQ (Priority Queuing, prioritní fronty), CQ (Custom Queuing, vlastní fronty), WRR (Weighted Round-Robin, vážený Round-Robin – cyklická obsluha) [56].

V mechanismu WFQ je ukládán každý tok do vlastní fronty, přičemž tok je v tomto případě definován protokolem – portem (TCP/UDP) – spolu se zdrojovou a cílovou IP adresou. Z adres a portu se při vstupu paketu vypočte haš, který současně plní funkci identifikátoru fronty. Pakety jednoho toku mají haš adres a portu stejný. Každému toku je konfigurací stanovena tzv. „váha". Podle váhy je frontám přidělena šířka pásma. Mechanismus WFQ nedostatečně chrání fronty s nižší prioritou – umožňuje, aby toky s vyšší prioritou zcela vytížily přenosové pásmo, což se negativně projevuje na průchodnosti toků s nižší prioritou sítí.

V situacích, kdy jsou softwarové i hardwarové fronty zaplněny, se v rámci WFQ uplatní algoritmus zahazování „Modified Tail Drop" [21], [56], viz kap 1.3.4. Pokud je použit, umožňuje algoritmus WFQ (v Cisco implementaci) vytvoření až 4 096 front nad každým rozhraním – velké množství front je třeba, každý tok potřebuje vlastní frontu [56].

Mechanismy WFQ, PQ a CQ byly nahrazeny moderními nástroji a mechanismy, kterými jsou CBWFQ (Class-Based Weighted Fair Queuing, třídní vážené fronty) a LLQ (Low-Latency Queuing, fronty s nízkým zpožděním) [56]. Tyto mechanismy umožňují stanovit minimální

šířku pásma, která je dané frontě garantována. To brání, aby toky s vysokou prioritou zcela vytížily linku provozem na úkor toků s prioritou nižší. Pokud je některá z front prázdná, v obvyklé konfiguraci se rovnoměrně rozdělí jí přidělené přenosové pásmo mezi ostatní fronty. Oba mechanismy WFQ i CBWFQ používají ve stavu přetížení, spolu s dalšími (viz kap. 1.3.4), mechanismy zahazování – „Tail Drop“ a „Modified Tail Drop“, v rámci CBWFQ lze navíc použít i WRED (Weighted Random Early Detection, vážená náhodná včasná detekce) [56].

Označení CB (Class-Based, tedy založený na třídách), ve zkratce mechanismu CBWFQ, odkazuje na to, že mechanismus využívá tříd provozu. Provoz může být do tříd zařazován podle různých pravidel, viz kap. 1.2.5. CBWFQ může pracovat, na rozdíl od WFQ, nejen s adresami a porty (reprezentovanými hašem), ale je možné využít veškerých možností identifikace paketů, které poskytují mechanismy klasifikace paketů a jejich přiřazování do tříd, včetně práce s protokoly vyšších vrstev, popsané v kap. 1.2.5.

V rámci CBWFQ (i LLQ) lze v Cisco zařízeních definovat maximálně 63+1 front (tříd) [21]. V běžné firemní síti lze identifikovat do 30 typů provozů. Z toho provoz, který je vhodné vložit do samostatné prioritní třídy, tvoří obvykle méně než 20 % [63]. Tento poměr se většinou promítá do konfigurace sítě obecně, ale v různých sítích lze pozorovat i výrazné odchylky.

Každá třída může v rámci CBWFQ definovat pouze 1 frontu. Proto jsou (v tomto kontextu) označení třída a fronta chápána jako rovnocenná. K 63 frontám je navíc přidána 1 softwarová fronta FIFO (označovaná „class-default“), do které jsou řazeny pakety, které nelze na základě tříd (pravidel) zařadit do jiných front.

V kap. 1.3.2 byl zmíněn plánovač a plánování v rámci jedné nebo všech front. Na většině Cisco zařízení (literatura [21] a [56] zmiňují jedinou výjimku – řadu směrovačů Cisco 7500) se pro plánování v rámci jedné fronty používá v mechanismu CBWFQ výhradně mechanismus FIFO, pouze fronta „class-default“ umožňuje volit z obou mechanismů, tedy FIFO i WFQ. Zmíněný směrovač Cisco 7600 umožňuje použít mechanismy oba [21]. V případě plánování v rámci všech front navrací plánovač informaci o tom, jakou část celkové šířky pásma rozhraní bude daná fronta využívat a které jí bude současně garantováno [21].

V rámci CBWFQ jsou pro fronty definovány dva limity – prvním je maximální rezervovatelná šířka pásma daného rozhraní, druhým limitem je šířka pásma dané fronty. Jednotlivým frontám lze ve standardní konfiguraci přidělit maximálně 75 % maximální rezervovatelné šířky pásma rozhraní. Zbývajících nerezervovatelných 30 % je dedikováno režii sítě. Nealokovatelnou 30% šířku pásma lze zmenšit až k 0, ale tento postup není doporučen [21].

Druhým ze soudobých moderních mechanismů front je mechanismus LLQ, který byl v předchozím textu již okrajově zmíněn. Mechanismus LLQ byl zaveden spolu s CBWFQ. LLQ nahrazuje starší mechanismus PQ, který, podobně jako WFQ, působil problémy datovým tokům s malou prioritou. LLQ nepracuje (na rozdíl od CBWFQ) s šířkou pásma, ale prioritou paketu. V LLQ lze, stejně jako v případě CBWFQ, definovat maximálně 64 front a je rovněž zajištěno, aby fronty s nižší prioritou nestrádaly na úkor front s prioritou vyšší. V LLQ je oproti CBWFQ navíc definována „fronta s nízkým zpožděním“. Fronta s nízkým zpožděním je obsluhována vždy jako první, tzv. strict priority (SP).

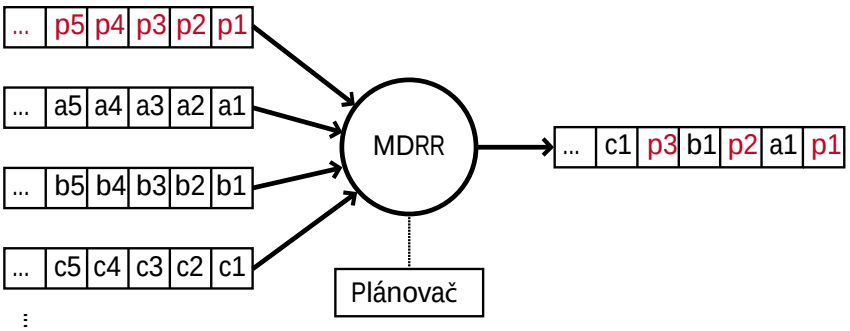
V případě mezního zatížení aktivního prvku jsou sice pakety náležící do „fronty s nízkým zpožděním“ obsluhovány prioritně, tak aby jim bylo garantováno deklarované nízké zpoždění, ale ty mohou být zahozeny a v datovém toku mohou být místo nich odeslány pakety z front s nižší prioritou. Aktivní prvky jsou téměř výhradně konfigurovány tak, aby do „fronty s nízkým zpožděním“ byly ukládány pakety VoIP s prioritou DSCP EF (viz kap. 1.2.5), kde je stanovena nejvyšší pravděpodobnost zahození. Při použití LLQ lze v dané třídě použít pouze mechanismus zahazování „Tail Drop“, WRED použít nelze [56].

Jedním z mechanismů, kterými je zajištěna ochrana front s nižší prioritou před strádáním, je kontrola shluků. Na Cisco prvcích je výchozí konfigurací stanovena povolená maximální velikost shluku paketů (burst) na 20 % šířky pásma [21]. Shlukem se v tomto případě rozumí množina po sobě odesílaných paketů náležící do stejné fronty (v případě linky o rychlosti 64 kb/s tedy shluk nesmí být větší než 1,6 kB).

Pozn. Strádáním front se myslí ponechávání rámců/paketů v dané frontě delší dobu, než je vhodné pro dodržení odpovídajících požadavků na parametry QoS, který je do fronty ukládán.

Zajímavým mechanismem je algoritmus MDRR (Modified Deficit Round-Robin, modifikovaný deficitní okruh) [21]. MDRR je relativně jednoduchý algoritmus, je t. č. implementováno na Cisco CRS směrovačích, ale i řadě 12000, [21], [64]. MDRR umožňuje použití 8 front, z toho 1 prioritní (7+1). Pro algoritmus je typické, že může pracovat ve 2 režimech – Strict priority mode (striktní priorita) a Alternate mode (alternující). Odesílání paketů z front je podobné jako v případě regulačního algoritmu „Token Bucket“, viz kap. 1.2.7. Plánovač pracuje s tzv. QV (Quantum Value, hodnota kvanta, kvantum), tedy stanoveným množstvím B, které je povoleno v daném okamžiku odeslat. Protože mají pakety ve frontách proměnlivou délku, ale paket je třeba odeslat celý (aby nebyl odeslán paket poškozený), je třeba odeslat více bajtů než je stanovené QV. Rozdíl mezi skutečně odeslaným počtem bajtů a QV je tzv. deficit, který je v dalším cyklu od QV odečten – průměrně je tedy odesláno množství bajtů odpovídající QV.

Fronty jsou v rámci MDRR obsluhovány cyklicky – z každé ze 7 prioritních front je postupně odesláno zmíněné QV. Výjimkou jsou pakety uložené v prioritní frontě, které jsou odesílány v režimu striktní priority – vždy za sebou, bez přechodu k odesílání QV z ostatních front. V režimu alternujícím jsou data QV odesílána plánovačem střídavě vždy prvně z prioritní fronty a následně z jedné z ostatních front, které jsou i nadále cyklicky střídány, viz obr. 1.11. Algoritmus MDRR je jednoduchý a výkonný, ale zvlášť v režimu striktní priority může docházet ke strádání neprioritních front [21], [56].



Obr. 1.11: MDRR – režim alternující

Použití každého z vyjmenovaných mechanismů má svá specifika. Aktuálně nejčastěji nasazovaný je mechanismus LLQ a kvůli jednoduchosti FIFO. Popsané mechanismy mohou, při nesprávné konfiguraci nebo nevhodném použití způsobit jitter, vytvářet shluky, zvyšovat zpoždění, některé fronty mohou strádat atd.

1.3.4 Mechanismy zahazování paketů

Zahazování paketů v aktivních prvcích je nutností v případech, kdy jsou přeplněny hardwarové a pokud jsou zavedeny i softwarové fronty. K přeplnění front dochází za situací, kdy je tok paketů na vstupu aktivního prvku vyšší (rychlejší) než odesílání na výstupním rozhraní. Krát-

kodobě lze k vyrovnání vyššího vstupního datového toku použít softwarové a hardwarové fronty, viz kap. 1.3.1, 1.3.2 a 1.3.3. Při dosažení stanovené úrovně naplněnosti front je třeba nadbytečné pakety „zahodit“. Důsledkem takového postupu je vznik shluků (burst) paketů v síti. Pokud je TCP paket zahozen, je to indikováno absencí potvrzení (SYN, ACK).

Kromě zahazování rámců/paketů mohou v případě zahlcení provozem aktivní prvky protokolem ICMP (Internet Control Message Protocol) zaslat odesílateli žádost o zpomalení odesílání (ICMP typ 4, kód 0, source quench – žádost o snížení rychlosti odesílání) – to lze použít jen v případě protokolu TCP. V případě UDP se tento ICMP kód nevyhodnocuje.

Nejjednodušší je již zmíněný mechanismus „tail drop“ (zahazování posledních), při jehož užití se zahazuje poslední paket nejdelší fronty. Metoda zahazování „Tail Drop“ je implementačně jednoduchá, její velkou nevýhodou je následná reakce sítě na zhození paketů. Protože jsou pakety přenášeny často ve shlucích, může být zahozením „přesahujících“ paketů zahozena příliš velká část provozu jednoho odesílatele, který v reakci na to sníží v souladu s doporučeními RFC 793 [65], RFC 3168 [32] a RFC 5681 [66] rychlost odesílání. Přímým důsledkem je pokles výkonu sítě.

Variantou algoritmu „Tail Drop“ je algoritmus „Modified Tail Drop“. Ten je možné nakonfigurovat, resp. použít dvěma způsoby. Prvním je algoritmus „early dropping“ (časné zahození), kdy je paket zahozen dřív, než je zaplněna daná fronta. Zejména v případě TCP protokolu se jedná o výhodné chování, protože vede ke zpomalení odesílání TCP paketů zdrojem provozu. Druhým algoritmem je „aggressive dropping“ (agresivní zahození), při jehož použití se pakety zahazují až při zcela plné frontě. V obou případech se zahazuje vždy poslední paket nejplnější fronty. Přitom se ale uvažuje ne počet paketů, ale jejich úhrnná velikost.

Kromě metody „Tail Drop“ bylo vyvinuto několik efektivnějších metod, mj. algoritmus RED (Random Early Detection, náhodná včasná detekce). Část paketů je zahozena už v okamžiku částečného zaplnění fronty. S rostoucím zaplňováním fronty roste počet zahozených paketů. Metoda zahazování paketů RED byla navržena, aby problém snižování výkonu sítě řešila [67].

Matematický model RED vychází zčásti ze vztahů popisujících Markovovy řetězce. Fronta je de facto homogenní a konečný Markovův řetězec s diskrétním časem.

Pozn. Markovovy řetězce jsou procesy, jejichž stavy jsou diskrétní. Pravděpodobnosti, že se systém v daném čase nachází v daném stavu, závisí jen na stavu systému v předchozím kroku. Markovovy řetězce lze popsat přechodovou maticí **P** (maticí pravděpodobností přechodu). Přechodová matice reprezentuje systém vždy v jednom kroku. Kroky jsou diskrétní. Obvykle se přechodová matice, která je čtvercová, zapisuje tak, že do řádků matice se zapisují pravděpodobnosti se vztahem k minulému stavu a do sloupců ke stavu budoucímu. Součet prvků (pravděpodobností) v řádku je roven 1 – řádek pokrývá všechny pravděpodobnosti. Některá literatura (např. [6]) zapisuje a používá matici přechodu oproti popsanému transponovanou. Přechodovou matici pro krok k lze určit jejím umocněním, tedy určením matice $\mathbf{P^k}$,[68], [69]. Každý z řádků přechodové matice je tzv. pravděpodobnostní vektor $\vec{s}$ [68], [69]. Protože je matice **P** stochastická a pravděpodobnostní vektor $\vec{s}$ je z ní odvozen, platí, že součet všech jeho prvků (n odpovídá počtu prvků) je roven 1, (1.12), [6].

$$\sum_{i=0}^{n} s_i = 1 \tag{1.12}$$

Aktivní prvek na základě výpočtu odhadu vývoje zaplnění fronty (založeno na Markovových řetězcích) rozhoduje, zda bude příchozí paket zahozen, nebo mu bude změněna priorita.
Mechanismus RED je popsán veličinami:

- h_o – okamžitá hloubka fronty, vyjadřuje množství dat ve frontě [21]. Hloubku fronty měří aktivní prvek trvale, resp. v každém časovém kroku a pro každou frontu odděleně.
- h_{min} – spodní prahová hodnota (konstanta stanovená uživatelsky).
- h_{max} – horní prahová hodnota (konstanta stanovená uživatelsky).
- h_c – celková hloubka (velikost) fronty.
- h_p (paket) – průměrná hloubka fronty je aktivním prvkem vypočítávána opakovaně v každém časovém kroku. Je definovaná vztahem (1.13), [6].
- d – pravděpodobnost zahození paketu, $d \in \langle 0; 1 \rangle$.
- p_z – pravděpodobnost přeznačkování paketu.
- Δc_p – počet paketů zařazených do fronty od posledního paketu označeného k zahození.
- s_i – i. prvek pravděpodobnostního vektoru $\vec{s}$.

$$h_p = \sum_{i=0}^{h_c} (i \cdot s_i) \tag{1.13}$$

Pravděpodobnost zahození d (přeznačkování p_z) je závislá na hloubce fronty a stanovené spodní a horní prahové hodnotě [67]. Vzájemné relace popisují vztahy (1.14) až (1.18).

- Pokud je ve frontě dost místa na uložení paketů, není třeba žádné pakety zahazovat ani přeznačkovat, viz vztah (1.14), [6], [67].

$$h_o < h_{min} \Rightarrow d = 0 \tag{1.14}$$

- Vztah (1.15) popisuje situaci, kdy ve frontě zbývá místo o velikosti $h_c - h_o$, ale pro zachování dobré propustnosti sítě je výhodné zahazovat 100 % příchozích paketů, tedy $d = 1$. Do stavu, kdy $h_o = h_c$ by se při použití algoritmu RED měla fronta dostat jen výjimečně, [6], [67].

$$h_o > h_{max} \Rightarrow d = 1 \tag{1.15}$$

- Pokud platí (1.16), řídí se zahazování rovnicemi vztahy (1.17) a (1.18), [6], [67].

$$h_o \in \langle h_{min}; h_{max} \rangle \tag{1.16}$$

Průměrná hloubka fronty je porovnávána se stanovenými limity (minimálním a maximálním prahem), na kterých záleží nakládání metody RED s pakety. Se zaplňováním fronty roste počet zahozených paketů.

$$d = \frac{h_p - h_{min}}{h_{max} - h_{min}} \tag{1.17}$$

Pravděpodobnost označení paketu (1.18) je odvozena od vypočtené pravděpodobnosti zahození (1.17), [6].

$$p_z = \frac{d}{1 - \Delta c_p \cdot d} \tag{1.18}$$

Dalším známým algoritmem (metodou) je WRED (Weighted Random Early Detection, vážená náhodná včasná detekce) [21]. Princip fungování je odvozen od principu zahazování paketů metodou RED, který doplňuje o váhy, tedy vazbu na diferencované služby, resp. DSCP AF, viz kap. 1.2.2. Různým prioritním třídám paketů je dána odpovídající pravděpodobnost zahození. Obdobně jako v případě RED jsou definovány horní a spodní prahová hodnota. Spodní prahová hodnota (v Cisco terminologii „Minimální prahová hodnota" [21]) je v doporučeních Cisco odvislá od DSCP a v rámci doporučení se pohybuje v intervalu $h_{min} \in \langle 33; 37 \rangle$. Např. pro DSCP AFx1 je doporučeno $h_{min} = 33$, pro DSCP EF je stanoveno $h_{min} = 37$, ale pro IPP 0 $h_{min} = 20$. Horní prahová hodnota (v Cisco terminologii „Maximální prahová hodnota") je doporučena ve všech případech $h_{max} = 40$ [21].

Platí, obdobně jako v případě RED, že pokud je hloubka fronty menší nebo rovna spodní prahové hodnotě, nezahazují se žádné pakety. Pokud je průměrná hloubka fronty větší než horní prahová hodnota, zahazují se všechny pakety. Pokud se průměrná hloubka fronty pohybuje v rozmezí minimálního a maximálního prahu, tedy $h_p \in (h_{min}; h_{max}\rangle$, zahazuje mechanismus pakety podle vztahu (1.19). Mechanismus zahazování WRED tímto způsobem brání neefektivnímu využití dostupné šířky pásma, podobně jako mechanismus RED. Konfigurací se stanovuje pro každou prioritní frontu v rámci WRED, kromě minimální a maximální prahové hodnoty, parametr MPD. MPD (Mark Probability Denominator) – „jmenovatel pravděpodobnosti označení" [21]. MPD se obvykle pohybuje v intervalu od 10 do 20. Ze vztahu (1.19) je zřejmé, že čím vyšší je MPD, tím je počet zahozených paketů nižší.

$$\frac{1}{MPD} = d \tag{1.19}$$

Konkrétní paket, který je zahozen, volí WRED (podobně jako RED) náhodně. Nastavení minimální prahové hodnoty odráží pravděpodobnost zahození danou polem DSCP. Platí, že čím je pravděpodobnost zahození paketu v DSCP poli vyšší, tím je spodní prahová hodnota obvykle nastavena na nižší úroveň (paket je zahozen dříve, resp. při menším zaplnění fronty, při menší průměrné hloubce fronty) a současně bývá upravena hodnota MPD tak, aby bylo zahazování častější (tedy menší MPD) [21], [56].

Kromě WRED je dalším známým algoritmem založeným na RED algoritmus FRED (Fair Random Early Detection, spravedlivá náhodná včasná detekce), který je jednodušší než RED. FRED pakety zahazuje náhodně při dosažení stanoveného intervalu $\langle h_{min}; h_{max}\rangle$, ale s konstantní pravděpodobností. Dalšími známými algoritmy jsou RIO, SRED (Stabilized Random Early Detection, stabilizovaný RED) – snaží se o stabilizaci obsazenosti front, LQD (Longest Queue Drop, zahazování z nejdelší fronty) – předpokládá, že nejvíce zatěžující zdroje, resp. typy provozu mají nejdelší frontu [6], [55].

Dalšími algoritmy, které se používají, jsou např. časté modifikace algoritmu Tail Drop např. WTD (Weighted Tail Drop, vážené zahazování posledních). Algoritmy Tail Drop i WRED, jsou mj. používány v aktivních prvcích Cisco. Algoritmus WRED je dostupný na většině Cisco prvků.

Ze vztahů (1.13) až (1.19) je patrné, že popsané algoritmy vyžadují určitý nezanedbatelný počet kroků. To je jednou z příčin, proč jsou stále často používány jednoduché FIFO, resp. Tail Drop mechanismy zahazování.

1.4 Řízení a architektura aktivních prvků

Kromě správy front, regulace a tvarování je pro QoS v aktivním prvku důležitý jak celkový hardwarový výkon, tak také základní principy návrhu aktivního prvku.

1.4.1 Přepínače se sdílenou pamětí

Sdílená paměť je implementována obvykle do jednodušších přepínačů. U tohoto typu zařízení jsou pakety po vstupu zapisovány do paměti, následně přečteny a odesílány na výstupní port. Paměť de facto plní roli spojovacího pole. Zmíněný princip je relativně jednoduchý, ale je zatížen řadou problémů. Jedním z nich je problém šířky pásma paměti. Pokud má aktivní prvek 4 vstupní/výstupní porty, musí být pro průchod rámce aktivním prvkem provedeny alespoň 2 paměťové transakce. Rámec bude 1× do paměti zapsán a 1× přečten. Paměť tedy musí mít propustnost alespoň 8× (4×2) větší než je rychlost každé ze vstupních/výstupních linek. Potřebná rychlost paměti roste s počtem portů.

Problém potřebného počtu operací se sdílenou pamětí lze mj. řešit jejím zrychlením, resp. širokopásmovým přístupem k paměti. V současnosti se obvykle používají paměti DRAM, jejichž propustnost u malých aktivních prvků už nepředstavuje problém [70], [71].

Jednou z metod, jak zvýšit efektivitu využívání sdílené paměti a tím propustnost celého zařízení je střádání přicházejícího provozu ve vstupní frontě – posuvném registru – přímo na rozhraní, dokud se nezaplní celá buňka (viz kap. 1.3.1), ta je pak zapsána do paměti. Obdobně pak na výstupu mohou být rámce uloženy do výstupního posuvného registru a odtud postupně (na L1 RM ISO/OSI po bitech) odesílány. Tento princip ušetří, v případě širokopásmového (paralelního) zápisu, počet sériových paměťových operací [72].

Přepínače tohoto typu jsou buď dodávány jako samostatné prvky, ale často jsou integrovány do jednoduchých domácích směrovačů, xDSL modemů. Zřídka mají víc než cca 5–8 vstupně výstupních portů.

1.4.2 Přepínače se sběrnicí a sdílenou pamětí

Přepínače se sběrnicí a sdíleným CPU patří ke generačně nejstarším. Byly vyvinuty z přepínačů s nejjednodušším sdílením média – sběrnice nebo paměti. Tyto přepínače propojovaly vstupy s výstupy přes jednu společnou sběrnici. Nevýhodou tohoto řešení je, že v jednom okamžiku může komunikovat pouze jeden vstup s jedním výstupem. U tohoto řešení neexistuje paralelní zpracování. Po dobu přenosu jednoho rámce není možné přenášet jiný [72]. Toto řešení je z hlediska QoS nevhodné.

Nejstarší směrovače a přepínače byly navíc řízeny jedním sdíleným univerzálním CPU. Toto řešení současně bylo a stále je do určité míry výhodné – umožňuje snadnou tvorbu a modifikovatelnost řídicího softwaru. Univerzální CPU, ale v dané roli nemá obvykle výkon, kterého by při zpracování rámců dosahoval specializovaný NPU. Tato koncepce přepínače je problematická i tím, že každý rámec prochází před odesláním na výstupní rozhraní sběrnicí alespoň 2×. Důsledkem je snížení propustnosti, která je závislá na propustnosti sběrnice. Ta musí být n násobně vyšší, než je požadovaná propustnost aktivního prvku – závisí na počtu operací prováděných s paketem a skutečným počtem průchodů paketu sběrnicí. Primární příčinou vynuceného opakovaného průchodu paketu sběrnicí je CPU (obvykle na samostatné kartě) připojené na stejnou sběrnici [72].

1.4.3 *Přepínače se sběrnicí, sdílenou pamětí a více sdílenými procesory*

Aktivní prvky vzešlé z koncepce prvků se sběrnicí a sdílenou pamětí, viz kap. 1.4.2 odstraňovaly nejužší místo – sdílené CPU. Novější generace přepínačů se sběrnicí a sdílenou pamětí byla osazena více procesory, jejichž úkolem byla správa a řízení předávání rámců.

Tato koncepce mj. umožňuje, aby jeden z procesorů zpracovával předávání rámců mezi jedním vstupem a výstupem (např. mezi vstupem 1 a výstupem 3) a současně druhý procesor zpracovával předávání mezi jinými rozhraními (např. mezi 4 a 6), atp. Celkově toto řešení zvýšilo propustnost, resp. snížilo požadavky na výkon každého CPU [72].

Řešení s použitím více procesorů umožnilo použít jednodušší procesory a současně zvýšit propustnost. I toto řešení ale může být zdrojem problémů. Může působit doručování rámců mimo pořadí, resp. aktivní prvek musí tento problém řešit. Navzdory zvýšení výkonu zůstala sdílená sběrnice prvkem limitujícím počet a propustnost vstupních/výstupních portů a dalších částí na ni připojených.

1.4.4 *Přepínače s křížovým spínačem*

Spojovací pole postavené nad křížovým spínačem (crossbar) je v různých modifikacích velmi efektivní způsob propojování vstupních a výstupních portů aktivních prvků. Použití křížového spínače není nové, ale spíše aplikací klasické metody spojování známé z telekomunikační techniky.

Konstrukce přepínače využívajícího principů křížového spínače v nejjednodušší podobě obsahuje sadu 2N paralelních sběrnic. Jednu sběrnici na vstupním portu a jednu na výstupním. Pokud vstupní sběrnice budou horizontální a výstupní vertikální, vzniklá matice je křížový spínač [72].

Řešení potenciálně poskytne N násobné zrychlení oproti jednosběrnicovému řešení. Propojení každého vstupu na každý vstup může být řízeno samostatně. Všech N sběrnic může být použito pro přenos dat paralelně a ve stejný čas. Předpokladem je, že v daném okamžiku nebude na výstupní port odesílán datový rámec, tedy, že výstupní port bude volný. Tento problém je ale řešen jak na úrovní plánování, tak front.

Přepínače, založené na křížových spínačích, navrhované cca od roku 1995, používají integrované obvody NPU namísto univerzálního CPU. Jsou rychlejší a levnější než univerzální CPU. Nevýhodou je, že se ve většině případů nedají přeprogramovat, viz kap. 1.5.1, [72]. Přepínač s N vstupy a N výstupy má N^2 spínacích bodů (crosspoint), které jsou individuálně ovládány.

1.5 Blokové schéma přepínače

Na síťové prvky lze z konstrukčního hlediska pohlížet na různých vrstvách počínaje problematikou mikroelektroniky a konče návrhy protokolů. Pro řešení projektu a zejména pro simulaci vybraných procesů v aktivním prku, viz kap. 3 a dále, je zvolen pohled klíčových konstrukčních celků, jež mají vliv zejména na sledovanou problematiku QoS.

Cílová implementace nemusí být provedena v hardwaru. V současnosti existují a v průběhu celého popisovaného výzkumu s nastupující virtualizací serverového hardwaru vznikaly softwarové implementace mnoha aktivních prvků. Mj. např. Cisco Nexus 1000V, Brocede virtual switch (Virtual Fabrics) atd. Kromě nutnosti propojit virtuální servery byla příčinou vzniku těchto prvků i snaha sjednotit administraci fyzických a virtuálních síťových zařízení. Některé implementace, např. Brocade, kombinují virtuální řízení doplněné o fyzické rozšiřující karty do serveru postavené nad ASIC (Application Specific Integrated Circuit, pro-

cesory navržené pro plnění přesně specifikovaných úloh, hardwarově akcelerující náročné operace), bez výstupně/výstupních portů.

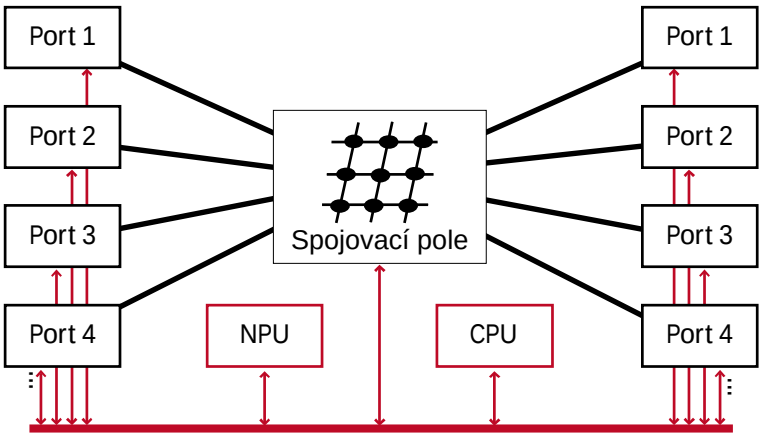
Současně s virtualizací aktivních prvků a růstem výkonu se projevila i snaha zvýšit univerzálnost aktivních prvků – výkon řídicí roviny, v současné době zejména na směrovačích umožňuje provozovat doplňkové služby. Typické jsou prvky Cisco ISR a další postavené nad univerzálním IOS XE, IOS XR. Zařízení jsou postavena nad Linuxovým jádrem, IOS je pouze démon. Do aktivních prvků je možné přidávat další funkce. Z popsaného je zřejmé, že předkládaná bloková schémata přepínače jsou pouze jednou z mnoha možností, jak může být přepínač navržen. Podrobně jsou různé architektury aktivních prvků popsány v [72], [73], [74], [75] atd.

Podle způsobu práce s pakety jsou aktivní prvky rozdělitelné na dvě skupiny. Prvky pracující s pakety až po jejich kompletním doručení jsou označovány jako „store-and-forward". Druhou skupinou jsou prvky, které zpracovávají paket již v průběhu jeho přijímání. Odesílání paketu je normálně možné zajistit až po přijetí hlavičky a kontrole jejích základních polí. Přidáním značek do hlavičky paketu, jakými jsou např. MPLS aj., lze na aktivním prvku ušetřit nutnost čekat s odesláním až do přijetí celého paketu. Nositelem informace o cíli paketu je vložená značka. Tato metoda zpracování se nazývá „cut-through" [76].

Zpracování „cut-through" využívají zejména MPLS přepínače a směrovače, které mohou ke směrování paketu využít zmíněné značky v jeho hlavičce. Dalším příkladem „cut-through" přepínání jsou hlavičky (pole) VLAN IEEE 802.1Q [24], [76]. Pokud je adresát paketů připraven v daném okamžiku přijímat paket, metoda „cut-through" jednoznačně předávání paketů zrychluje. Průběžné odesílání přidává (ve srovnání se zpracováním „store-and-forward") do přenosu pouze minimální jitter. Nevýhodu metody „cut-through" je, že je-li paket poškozen, je alespoň z části odeslán. V případě metody předávání „store-and-forward" je to vyloučeno [76].

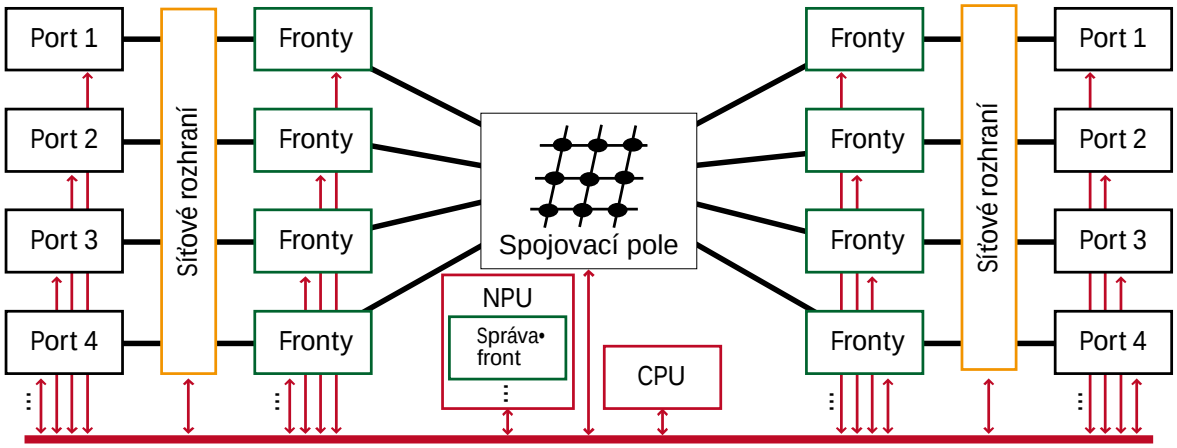
Jednoduché schéma čtyřportového přepínače je zachyceno na obr. 1.12. Schéma na obr. 1.12 odděluje vstupní a výstupní porty do samostatných bloků. Oddělení je obvyklé i v praxi, kdy jsou na vstupních a výstupních portech prováděny s rámci rozdílené operace. Jednotlivé významné součásti přepínače jsou popsány v kap. 1.5.1 až 1.5.4. Černě vyznačené cesty odpovídají datové rovině přepínače, červeně je vyznačena řídicí (kontrolní) rovina. Napájecí obvody, konfigurační rozhraní, paměti atd. nejsou ve schématu zachyceny.

Schémata obr. 1.12 a obr. 1.13 jsou cílena na řešenou problematiku. Architektura zařízení osazená několika linkovými kartami a řídicí kartou (supervizorem) je principielně obdobná, ale konstrukčně výrazně složitější. Takovým zařízením je např. přepínač Cisco 6500, který je určen jako agregační páteřní přepínač pro telekomunikační operátory, případně do kampusových sítí jako centrální prvek. Architektura supervizoru Cisco 6500 viz [77].



Obr. 1.12: Zjednodušené blokové schéma přepínače

Schéma z obr. 1.12 je na obr. 1.13 rozšířeno o prvky se vztahem k řešené problematice – bloky front a bloky síťového rozhraní. Prakticky je problematika front obvykle řešena v rámci NPU, viz kap. 1.5.1. Jednotlivé porty jsou obvykle prezentovány jako součást bloku síťového rozhraní. Ve vztahu k řešené problematice je ale použité zakreslení výhodnější.



Obr. 1.13: Blokové schéma s důrazem na řízení správy front

1.5.1 SÍŤOVÝ PROCESOR – NPU

Síťový procesor – Network Processor Unit je klíčovým řídicím prvkem přepínačů. V prvních generacích byla problematika přepínání, ale i směrování, řízena procesory GPP (General-Purpose Processor, tedy procesory pro obecné použití), např. v Cisco 2500. V soudobých zařízeních jsou ale NPU většinou procesory ASIC [55]. Existují varianty bez možnosti rekonfigurace i varianty řízené firmwarem. Známí výrobci aktivních prvků (Cisco, Juniper, HP atp.) se obvykle starají o implementaci NPU do konkrétního výrobku a o tvorbu softwaru. Architektury konkrétních procesorů se mnohdy výrazně liší. Vývojem síťových procesorů se zabývají, kromě jiných, např. IBM, Motorola, Intel. Nejznámější je pravděpodobně Broadcom.

Úkolem NPU je hardwarově akcelerovat úkoly, jejichž zpracování běžnými GPP by omezovalo propustnost aktivního prvku. Konkrétní funkce jsou dány konkrétním typem NPU. Obecně ale mezi základní funkce patří:

- řízení přístupu k médiu MAC (Media Access Control),
- zpracování paketů – extrakce polí, zpracovávání hlaviček, šifrování,
- klasifikace paketů,
- regulace a tvarování provozu.

V jednom aktivním prvku může být více NPU, resp. dalších ASIC. Běžně se 2 a více ASIC vyskytuje i v relativně levných zařízeních – např. přístupový přepínač Cisco Catalyst 3850, kde každý ze 2 ASIC je určen pro určitou skupinu portů. Přitom každý s jeho ASIC má propustnost 82 Mp/s

Architektura a návrh NPU je komplexní problém. Jedním z mnoha příkladů NPU může být např. Intel IXP1200. Jádro NPU je tvořeno šesti RISC procesory (terminologií Intel microengines). Ty jsou určeny pro zpracování datové roviny. Dále obsahuje NPU ARM jádro (tzv. StrongARM core) pracující na 166 MHz, které slouží k plánování a poskytuje další obslužné a servisní funkce. NPU obsahuje řadič externí SDRAM paměti (64b), řadič SRAM (32b), sběrnici IX sloužící pro přenos dat do a z NPU (64b, 104 MHz s propustností 6,6 Gb/s), [78].

1.5.2 CPU A MANAGEMENT

Univerzální procesor v aktivním prvku patří stejně jako NPU (viz kap. 1.5.1) do řídicí roviny. Úlohou univerzálního procesoru je podporovat běh veškerých úloh, pro které není v aktivním prvku dedikován ASIC. Mj. se jedná o správu aktivního prvku, běh operačního systému (IOS, Junos, Linux). Dále s tím spojený management konfigurací atd. Kromě toho také CPU plní některé úkoly spojené s primárními úkoly aktivního prvku. Většinou se jedná o ty, jež není nutné plnit tak rychle, aby jejich opožděné plnění snižovalo propustnost celého aktivního prvku (v případě směrovačů např. naplňování a správa směrovacích tabulek).

1.5.3 SÍŤOVÉ ROZHRANÍ

Blok síťového rozhraní pracuje s protokoly druhé vrstvy RM ISO/OSI, spoluvytváří „kontext“ paketu. Mj. také zajišťuje šifrování na 2. vrstvě RM ISO/OSI – MACSec (norma 802.1AE [50]).

V případě Cisco prvků se jedná o tzv. Cisco TrustSec (CTS). Při odesílání provozu síťové rozhraní naopak zapouzdřuje protokol 3. vrstvy RM ISO/OSI do protokolu 2. vrstvy atd.

Pozn. Norma IEEE 802.1AE (standardizovaná v roce 2006) [50] standardizuje šifrování a další bezpečnostní mechanismy na 2. vrstvě RM ISO/OSI. Využití technologie vyžaduje implementaci jak na straně koncových zařízení, tak na straně síťových prvků. Při nasazení standardu 802.1AE je šifrovanou komunikaci před zpracováním v aktivním prvku třeba dešifrovat a před odesláním do dalšího segmentu sítě znovu zašifrovat na výstupním síťovém rozhraní. Podpora 802.1AE bývá obvykle hardwarová, což znamená minimální nárůst zpoždění v chráněné síti [77]. Implementace šifrování na 2. vrstvě je náročná na optimalizaci, přesto byla do aktivních prvků implementována de facto hned po standardizaci (od r. 2006) [79].

1.5.4 SPOJOVACÍ POLE

Spojovací pole je centrální strukturou přepínačů a náleží do datové roviny aktivního prvku. Je jednou z periférií NPU. Bývá implementováno v rámci tzv. přepínacího procesoru (switch chip). V literatuře používaný pojem „switch fabric“ je buď označením právě pro spojovací pole, ale často je „switch fabric“ chápána komplexně a zahrnuje ještě mj. správu vyrovnávacích pamětí, fronty, regulaci a tvarování [76].

Nejznámější a nejjednodušší typ spojovacího pole je křížový spínač (crossbar), viz kap. 1.4.4. Další známé topologie spojovacího pole jsou mnohovrstvé propojovací sítě (Multistage Interconnection Networks) – např. Closovy sítě, Benes, Cantor, Batcher–Banyan. Další příklady jsou Generalized-Cube Network (GCN), Augmented Data Manipulator Network (ADMN), Improved Logical Neighborhood (ILN) a další (např. topologie založené na časovém multiplexu TDMA) [6], [76].

Pozn. Charles Clos představil Closovy sítě v r. 1953, výzkum možností vylepšení plánování probíhá trvale – použití přepínacích algoritmů Matching Algorithms for Clos (MAC) [80], algoritmus Concurrent Round-Robin v Clos síti [81] atd.

U spojovacího pole je rozlišeno, zda je jeho struktura blokující či neblokující. Neblokující struktura je taková, u které nemůže dojít k tzv. vnitřnímu blokování, tedy stavu, kdy v daném okamžiku používání některé jeho části brání v průchodu dalších rámců. Naopak struktura, kde k vnitřnímu blokování dochází – tedy kdy dochází k „soutěži" o zdroje ve spojovacím poli, se nazývá blokující. Absence vnitřního blokování by měla být garantovaná pro všechny procházející typy rámců [76]. Blokování lze předcházet návrhem kombinujícím rychlost spojovacího pole a správně navržené vyrovnávací paměti (správnost jejich návrhu je důležitá – paměť bude vnášet do procesu zpoždění). Podrobně jsou různé typy spojovacích polí analyzovány v [6], [55], [76].

1.5.5 Průchod rámce přepínačem

Průchod rámce aktivním prvkem v souladu s blokovým schématem, viz obr. 1.12 a kap. 1.5 lze shrnout do následujících 7 kroků.

1. Rámce, které vstoupí do každého z n vstupně-výstupních portů, jsou hned po vstupu zkontrolovány na výskyt problému se vztahem ke 2. vrstvě RM ISO/OSI – kontrola CRC (Cyclic Redundancy Check), kontrola MAC adres atd.
2. Pokud je použito šifrování podle 802.1AE, je na příslušném ASIC (v případě Cisco, CTS ASIC) vstupující provoz dešifrován. Dále je extrahován SGT (Security Group Tag), určený RBACL (Roles-Based ACL) a provoz je předán k dalšímu zpracování.
3. ASIC v rámci síťového rozhraní uloží paket do paměti, zpracuje se hlavička paketu – odchozí port, VLAN podle 802.1Q nebo ISL, viz kap. 1.2.1.
4. Paket je předán do „fabric interface", tedy vstupu spojovacího pole a současně je paketu v ASIC vytvořen kontext. Informace z hlavičky paketu jsou odeslány do NPU. V případě kampusového přepínače Cisco 6500, který může být osazen několika linkovými kartami, je rámec předán do PFC4 (Policy Feature Card 4. generace, součást supervizoru 2T. Obsluhuje předávání paketů IPv4 a IPv6, multicast, MPLS a zpracování protokolů 2. vrstvy. Kromě předávání paketů je PFC4 také zodpovědná za zpracování ACL, aplikaci politik regulaci, tvarování atd.) Kromě PFC4 může být rámec nasměrován v Catalystu 6500 do DFC4 (Distributed Forwarding Card 4), což je obdoba PFC4 integrovaná na linkové kartě – rámec není třeba přes vnitřní sběrnici (back plane) předávat do supervizoru a zpět [77].
5. Po zpracování 2. a 3. vrstvy je předán výsledek do vstupu spojovacího pole a případně replikačního ASIC.
6. Následně projde rámec spojovacím polem a je v případě potřeby v replikačním ASIC replikován. Replikace se využívá buď v případě multicastu nebo pro potřebu monitoringu – SPAN (Switched Port Analyzer) [82].
7. Paket je předán výstupnímu portu (pokud je cíl lokální) nebo prostřednictvím supervizoru jiné linkové kartě a odeslán.

Pozn. Využití přepínání paketů přímo na linkových kartách podstatně zvyšuje propoustnost aktivního prvku. Ve zmíněném případě Catalytu 6500 je maximální výkon šasi se supervizorem 2T 60 Mp/s (na IPv4). Každá DFC4 přidá dalších 60 Mp/s souhrnné kapacity předávání paketů na šasi [77].

1.6 SOUVISEJÍCÍ PUBLIKACE

Kromě prací, resp. literatury, průběžně odkazovaných v celém textu byl výzkum inspirován následujícími publikacemi.

1.6.1 SÍŤOVÉ PRVKY A QOS

Článek „Wire-speed Traffic Management in Ethernet Switches“ [83] se zabývá hardwarovými architekturami umožňujícími na linkové rychlosti (1Gb Ethernet) klasifikovat pakety. Autory jsou Mishra S., Guruprasad A., Hu Ch., a kol.

Samosměrovacími spojovacími poli se zabývá článek „A Self-routing Switch Fabric Architecture on a Chip“ [84]. Autory jsou Jang H. a Kim H. V článku jsou popsány jak již známé modely spojovacích polí – křížový spínač, samosměrovací pole (Beneova síť), paralelní architektury spojovacích polí, tak také nový návrh samosměrovacího pole.

1.6.2 SÍŤOVÉ PRVKY ŘÍZENÉ NEURONOVÝMI SÍTĚMI

Vladislav Škorpil a Jiří Šťastný v článku „Alternatives of Converged Network Control“ [85] popisují výsledky výzkumu alternativních síťových prvků využívající neuronové sítě. Článek porovnává jednotlivé neuronové sítě z hlediska výhodnosti jejich použití pro řízení aktivního síťového prvku.

Článek „Network Elements Controlled by Artificial Neural Network“ [86] publikovaný na konferenci WSEAS V. Škorpilem a D. Novákem se zabývá návrhem aktivního prvku řízeného neuronovou sítí. (Pozn. v článku prezentovaný model je od modelu vytvořeného v rámci předkládaného výzkumu odlišný).

V. Škorpil a S. Kamba představili v konferenčním příspěvku „Back Propagation and Genetic Algorithms for Control of the Network Element“ [87] genetické algoritmy a algoritmus – neuronovou síť se zpětným šířením chyby v použití na řízení síťového prvku.

Článek „A Neural Network Solution to QoS-IP Team-Optimal Dynamic Routing“ [88] autorů Baglietto M., Battistelli G., Bolla R. a kol. se zabývá problematikou dynamického směrování v paketových sítích v distribuovaných uzlech. Autoři využívají pro řešení problému neuronové sítě s dopředným šířením vstupního signálu tréninkového vzoru. Celý problém je řešen v síťovém simulátoru NS-2.

1.6.3 MODELOVÁNÍ NEURONOVÝCH SÍTÍ

Článek „Neural Networks Demo using MATLAB 6.5“ [89], jehož autorem je Michael Y. Jiang se zabývá modelováním XOR pomocí nástroje „Neural Network Tool (nntool), který je součástí Neural Network Toolboxu, součásti MATLABu 6.5. Autor popisuje jednoduchou metodu návrhu a tréninku neuronové sítě. Popisuje postup vytvoření sítě a její adaptaci. Nástroj nntool je rychle a snadno použitelný, problematičnost jeho použití je patrná až po delším užívání a podrobném seznámení s jeho možnostmi. Ačkoli je tento článek mnoha autory hojně citován, v něm zveřejněné výsledky se nepodařilo uvedeným postupem replikovat, viz kap. 3.3.3.

Kniha „Neural Network Toolbox 7“ [90], jejímiž autory jsou Mark H. Beale, Martin T. Hagan a Howard B. Demuth je nejen referenčním manuálem, ale poskytla i řadu dalších cenných informací.

Technická zpráva „What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation“ [91], jejímiž autory jsou Lawrence S., Giles L., Tsoi Ch. se zabývá problematikou nastavení velikosti neuronové sítě – počtem potřebných neuronů v jednotlivých vrstvách.

1.6.4 *Virtual Network User Mode Linux*

Andreas Steffen, Eric Marchionni a Patrik Rayo popisují v článku „Advanced Network Simulation under User-Mode Linux“ [92] využití UML ve výuce. Článek je úvodem do použití UML jako testovací platformy.

David Fernández, Tomás de Miguel a Fermín Galán v článku „Study and Emulation of IPv6 Internet-Exchange-Based Addressing Models“ [93] popisují použití UML jako nástroje pro modelování, simulaci a testování přidělování IPv6 adres mobilním uživatelům.

Kniha „User Mode Linux“, kterou publikoval Jeff Dike [94], je přehledný a současně podrobný manuál k UML.

Článek „Use of Virtualization Tools in Computer Network Laboratories“ napsaný Fermínem Galánem, Davidem Fernándezem a dalšími [95] popisuje použití VNUML v laboratoři počítačových sítí, jako užitečného nástroje pro efektivní simulaci komplexních počítačových sítí.

Výkon virtuálních počítačů a dalších nástrojů je předmětem článku „Quantifying Performance Properties of Virtual Machine“ [96]. Autory jsou Xianghua Xu, Feng Zhou a Jian Wan Yucheng Jiang. Autoři srovnávají virtualizační nástroje (prostředí) Xen, KVM a VMware.

2 CÍLE DISERTAČNÍ PRÁCE

Primárním cílem, jenž stanovuje samotné téma práce, je návrh nového aktivního síťového prvku s novou strukturou a řízením neuronovou sítí, na rozdíl od běžně dostupných aktivních prvků řízených konvenčními algoritmy. Počítá se s integrovanou podporou QoS. Průnikem vytipovaných problémů aktivních prvků, viz dále, a tématu disertační práce jsou cíle disertační práce:

1. Analyzovat vybrané aktivní prvky – přepínače.
2. Zvolit vhodnou neuronovu síť pro řízení přepínače.
3. Navrhnout nový přepínač řízený neuronovu sítí s podporou QoS.
4. Navrhnout efektivně modelovatelný protokol.
5. Navržený přepínač modelovat a optimalizovat.

Inspirací pro stanovení cílů byly problémy identifikované při zpracování kap. 1 a po provedení měření, viz kap. 1.3.1, 3.1 a 4.1. Vyřešení těchto problémů by mohlo přispět k celkovému růstu oboru. Jedná se zejména o:

1. Problém chybné identifikace provozu a jeho zařazení do správného toku, zejména pokud se jedná o provoz koncových uživatelů (provoz různých nestandardních VoIP a videokonferenčních systémů, časté využívání HTTP službami, pro které nebyl navržen).
2. Problém doručování paketů mimo pořadí.
3. Problém výkonnosti velkých spojovacích polí při plné zátěži.
4. Potřeba reklasifikovat provoz na hraně sítě, viz kap. 1.2.
5. Problematika bezpečnosti.
6. Problematika energetické spotřeby aktivních prvků.

Z vyjmenovaných problémů si tato disertační práce klade za cíl zabývat se zejména řešením problémů definovaných v bodech 1–3, na jejichž základě byly definovány cíle disertační práce 1–5.

Nezbytným prvním krokem je provést analýzu v současné době používaných způsobů řízení aktivních prvků, jejich výkonnosti, algoritmů pro správu softwarových front, které s řízením provozu na výstupu aktivního prvku přímo souvisí. Bylo také třeba stanovit požadavky služeb na QoS s důrazem na VoIP, ale také obecně v konvergovaných datových a telekomunikačních sítích. Dalším z cílů bylo shrnout klíčové mechanismy a protokoly ovlivňující kvalitu služby v IP sítích. Mapování současného stavu konvenčně řízených síťových prvků je uvedeno v kap. 1.

Druhým a třetím z cílů je navrhnout přepínač řízený neuronovu sítí. K tomu je třeba vybrat vhodnou neuronovou síť a analyzovat vhodnost jejího použití k řešení stanoveného problému. Využití umělých neuronových sítí, jako jednoho z nástrojů umělé inteligence, se osvědčilo při řešení úloh, u nichž je obtížné algoritmizovat rozhodovací proces kvůli vysoké variantnosti možných vstupů a výstupů. Cílem je využít umělou neuronovu síť, jako efektivnější alternativu k v současné době používaným algoritmům řízení aktivního prvku. Integrovat řízení neuronovou sítí spolu s konvenčními algoritmy pro dosažení maximální přesnosti a rychlosti při rozhodování o cíli paketu, resp. rámce. Zajistit minimální latenci, upřednostnit provoz s vyšší prioritou a zabránit strádání provozu s prioritou nižší. Je třeba stanovit, které části aktivního prvku je vhodné řídit konvenčními algoritmy a pro které části řízení se lépe hodí použití zvolené neuronové sítě a to s důrazem na QoS.

Předchozí výzkum, zmíněný mj. v kap. 1.6.1, se problematikou aplikace neuronových sítí v řízení síťového provozu zabýval dílčím způsobem. Výzkum prokázal, že v dílčích aplikacích lze pro řízení síťového provozu umělou neuronovou síť použít.

Čtvrtým a pátým – konečným – cílem je vytvořit nový komplexní simulační model síťového prvku vybaveného novým typem řízení. Součástí je návrh technologií, které umožní samotný výzkum aktivního prvku a také prověření dosažených výsledků. Aktivní prvek může být buď modelován čistě softwarově, nebo může být provedena hardwarová implementace. Dominantní je softwarový model, v prvních fázích byla prověřována i možnost hardwarové implementace, kterou se stručně zabývá kap. 4.1.

Jakkoli je v rámci současných konvergovaných datových/telekomunikačních sítí nejpoužívanější protokol IP, pro použití v modelech se příliš nehodí. Proto je cílem i návrh nového protokolu, který ponese klíčové části zmíněného IP protokolu, ale bude pro modelování dostatečně jednoduchý a tedy umožní snadnou detekci případných chyb v navrhovaných strukturách a algoritmech. Současně by modely postavené nad novým protokolem měly umožňovat přenos poznatků i zpět do reálných sítí postavených nad IP protokolem. Důraz je dán zejména na implementaci polí se vztahem k průchodu paketu/rámce aktivním prvkem a také na implementaci polí se vztahem ke QoS.

Pro ověření navrhovaných řešení je třeba vytvořit model aktivního prvku umožňující sledovat zkoumané parametry. Cílem je, aby byl model přehledný a flexibilní. Je tedy nutné zvolit vhodné prostředí pro provedení simulací. Navrhovaný aktivní prvek bude primárně modelován v jazyce a softwaru Matlab a Simulink.

Pro model budou mj. navrženy nové generátory provozu – vytvořený model musí umožňovat paralelní generování paketů, tedy simulovat několik různých zdrojů síťového provozu přicházejícího na různé vstupní porty aktivního prvku. Dále musí umožnit individuální nastavení velikostí vstupních a výstupních zásobníků, modifikovat zpracování průchozího provozu ve výstupních frontách atd. Jako podklad pro dílčí rozhodnutí o typu zvolené architektury, neuronové sítě apod., vznikne i několik dalších dílčích modelů, zaměřených na konkrétní zkoumaný problém.

Vzhledem k charakteru řešené problematiky není třeba vytvářet kompletní fyzikální model. Cílem je návrh algoritmů, architektury a ověření teorie komplexním modelem. Hardwarová implementace je cílem budoucího výzkumu v rámci širší skupiny.

3 Analýza problému

Rozborem vytčených cílů bylo zřejmé, že vhodnější bude výzkum orientovat spíše na architekturu přepínačů než směrovačů. Přepínače svojí architekturou pokrývající maximálně oblast zájmu prováděného výzkumu. Moderní přepínače podporují i protokoly vyšších vrstev, viz kap. 1.4 a 1.5. Přepínače s podporou L3 RM ISO/OSI jsou často některými výrobci doporučovány i do rolí PE, viz kap. 1.2 (což obecně nelze doporučit kvůli často nedostatečné kapacitě směrovacích tabulek v přepínačích). Přesto se ale jeví výhodnější pohlížet na vyvíjený prvek jako na přepínač. Pro výzkum v souladu s cíli vytyčenými v kap. 2 je nezbytná práce s více porty. Pokud je uvažován přepínač s podporou L3, jsou problémy QoS, které musí vyhodnotit, prakticky totožné s problematikou vyhodnocovanou nativním směrovačem.

Problémy definované v rámci kap. 2 ve vazbě na řešenou problematiku se týkají zejména:

- celkové architektury přepínače (počtu portů, velikostí pamětí, umístění zásobníků atd.),
- architektury spojovacího pole,
- řízení spojovacího pole,
- správy front, plánování, a zahazování paketů.

V počáteční fázi výzkumu (r. 2008–2009) byly provedeny dílčí testy a analýzy možností. Bylo rozhodnuto o vývoji modelu aktivního prvku řízeného neuronovu sítí. Byly zvažovány a prakticky testovány 3 směry a možné implementace navrhovaného řešení. Počátek výzkumu byl volný, bez vazby na určitou platformu nebo software. Každý ze směrů určitým způsobem analyzoval a testoval možnosti dalšího postupu.

Pro vývoj aktivních prvků je třeba komplexní vývojové prostředí umožňující vývoj komplexních modelů počítačové sítě. Prostředí by mělo být flexibilní a výsledná simulace by měla umožňovat simulovat reálné situace. Pro tyto účely jsou používány simulační nástroje umožňující modelovat a analyzovat výkonnost sítě, její slabá místa. Takovými nástroji jsou např. OPNET Modeler nebo Network Simulator. Ačkoli obzvláště Network Simulator někteří autoři např. [88] pro vývoj aktivních prvků s využitím neuronových sítí používají, návrh řešení není dostatečně efektivní. Při vývoji s použitím neuronových sítí a genetických algoritmů se ale jeví jako dobrá volba MATLAB s relevantními toolboxy. Předprototypové testování by oproti základnímu výzkumu mělo probíhat v prostředí co možná nejpodobnějším reálným podmínkám, resp. ideálně přímo v reálném prostředí. Pro prototypový vývoj se naopak jeví jako ideální vývojové karty založené na FPGA. Problém takových vývojových karet je relativně obtížná implementace vytvořeného algoritmu.

Prvním směrem výzkumu byla emulace a testování aktivních prvků v rámci platformy VNUML. Tento výzkum probíhal zejména ve spolupráci s ESIEE Paris – Département informatique. Cílem bylo nahradit přepínač implementovaný v rámci platformy VNUML vlastním prvkem řízeným neuronovou sítí. Ta měla být v závěru implementována knihovnou Fast Artificial Neural Network Library, resp. OpenNN. Problematikou modelů vytvořených s využitím VNUML se zabývá kap. 4.1.

Druhá, poslední větev výzkumu spočívala v čistě simulačním řešení – model vytvořený v MATLAB Simulinku s využitím simulovaného provozu. Po prověření výše zmíněných směrů se použití MATLABu a na něj navázaných knihoven, ukázalo jako ideální a od r. 2009 probíhal výzkum prakticky pouze s použitím těchto nástrojů.

3.1 VÝKONNOST VYBRANÝCH AKTIVNÍCH PRVKŮ

Pro posouzení charakteristik vlastností vytvořených modelů (zejména v rámci práce na platformě VNUML) a srovnání jejich výkonnosti s komerčně dostupnými přepínači bylo provedeno několik sérií měření výkonnostních parametrů přepínačů. Konkrétně byly testovány přepínače Cisco Catalyst 2960, HP ProCurve 2626 (orientované do přístupové sítě, podobně jako uml switch, viz dále) a Micronet SP608K (méně výkonný přepínač – pro srovnání). Byla testována výkonnost virtuálního přepínače „uml switch", který je součástí UML (User Mode Linux), resp. VNUML (Virtual Network User Mode Linux). Systém VNUML byl použit jako jedna z perspektivních testovacích platforem, model viz kap. 4.1.

Topologie a použitý model VNUML je popsán v kap. 4.1. Výkonnost použitého modelu je ve srovnávacích testech uvedena souhrnně zejména proto, že byly provedeny až po vytvoření modelu VNUML. Kompletní výkonnostní testy posloužily k posouzení výkonnosti platformy VNUML, jejích vlastností a v konečném důsledku změnily směr výzkumu – vedly k odklonu od platformy VNUML, viz kap. 4.1.3.

Provedené výkonnostní testy byly založeny na doporučení RFC 2544 [97], které je v současnosti používáno pro tento typ testů nejčastěji (pozn.: v době, kdy byly testy prováděny, nebyla dokončena implementace doporučení ITU-T Y.1564 [98] do dostupných analyzátorů). Celé doporučení RFC 2544 nebo jeho části jsou implementovány do síťových testerů téměř všech výrobců.

Detailně je metodika testování zpoždění popsána v doporučení RFC 2544 [97] a v doporučení RFC 1242 [99]. Přepínače jsou zařízení typu store-and-forward [99], tedy taková, na kterých je zpoždění (latence), viz kap. 1.1.2, definováno jako doba, která uplyne mezi přijetím posledního bitu rámce vstupujícího do zařízení (t_{fi}) a okamžikem, kdy je odesílán první bit rámce na výstupním portu (t_{f0}) [99]. Symbolický zápis, viz rovnice (3.1).

$$l_a = t_{fo} - t_{fi} \tag{3.1}$$

Zpoždění bylo měřeno s použitím paketů velikostí 64, 128, 256,… až do 1 518 B, viz tab. 3.1. Každý test byl 20× opakován, hodnoty v tab. 3.1 jsou aritmetickým průměrem změřených hodnot. Např. zpoždění l_a bylo vypočteno podle vztahu (3.2), kde n označuje počet měření, $n = 20$, l_i odpovídá změřené hodnotě, i označuje pořadí měření.

$$\overline{l_a} = \frac{1}{n}\sum_{i=1}^{n} l_i \tag{3.2}$$

Test jitteru (proměnlivého zpoždění), viz kap. 1.1.3, je založen na změřených hodnotách zpoždění vždy pro každou testovanou velikost paketu. Samotný jitter je spočítán podle RFC 3550 [11] (kap. interarrival jitter) jako rozdíl doby příchodu po sobě jdoucích paketů. Jitter byl, podobně jako zpoždění, průměrován, viz (3.2) z 20 provedených měření.

3.1.1 TEST ZPOŽDĚNÍ

Testy zpoždění byly spuštěny mj. také na systému VNUML. Pro tyto potřeby byl použit testovací scénář „hvězda", viz obr. 4.2 a kap. 4.1.2. Jako referenční systém byl zvolen přepínač Cisco Catalyst 2960 řízený systémem IOS 12.2(35)SE5. Vyjma uvedených výjimek bylo celé měření provedeno síťovým testerem VeEX VePAL TX300/e, firmware 3.2.1.2, dále jen VeEX. Pro lepší kategorizaci výsledků byl do testů zahrnut také přepínač Micronet SP608K. Tento 8portový přepínač není uživatelsky konfigurovatelný a je primárně určen pro domácí sítě a malé kance-

láře. Dalším přepínačem, který byl zařazen do testů, byl HP ProCurve 262 (firmware H.10.83). Jedná se o levný přepínač určený do nenáročných provozů v rámci přístupové sítě. Jeho dokumentace deklaruje zpoždění menší než 12 μs [100].

Velikost paketu [B]	Zpoždění [μs]			
	VNUML	Catalyst 2960	Micronet SP608K	HP ProCurve 2626
64	129,90	17,86	69,08	74,38
128	132,10	22,76	94,68	96,18
256	131,80	28,26	145,88	146,92
512	130,70	39,16	248,24	248,44
1024	126,80	60,92	453,02	453,24
1280	131,30	72,42	555,44	555,62
1518	164,60	82,34	650,62	650,58

Tab. 3.1: Srovnání zpoždění

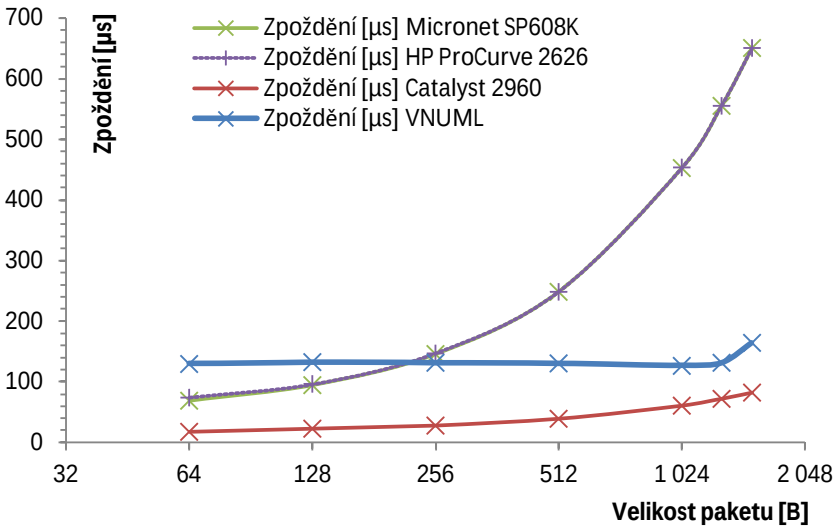
Výsledky testů jsou zachyceny v tab. 3.1 a na obr. 3.1. Zpoždění v simulačním prostředí VNUML bylo měřeno pomocí obecně známého programu Ping zasláním paketu „ICMP echo request," tedy s požadavkem na odezvu systému. Velikost testovacího paketu byla nastavována na hodnoty shodné s ostatními měřeními a doporučené v RFC 2544, tab. 3.1. Protože program Ping čeká na odezvu protilehlého systému (ICMP echo replay), je vnesené zpoždění de facto poloviční oproti zpoždění změřenému, l_a by tedy mělo být vypočteno podle vztahu (3.3). Navíc je měření postiženo ještě systematickou chybou – zpožděním vneseným zpracováním protokolu ICMP.

$$\overline{l_a} = \frac{1}{2n}\sum_{i=1}^{n} l_i \tag{3.3}$$

Pozn. Pro zajištění objektivity měření bylo provedeno srovnávací měření. Byla srovnána přesnost analyzátoru VeEX s měřením programem Ping. Zapojení při srovnávacím měření odpovídalo topologicky obr. 4.2. Centrum topologie tvořil Catalyst 2960. Koncové uzly byly tvořeny, v případě měření programem Ping, PC (parametry viz kap. 3.1.3) a v případě měření analyzátorem VeEX jeho měřicími porty. Měření byla vzájemně srovnána. Podle vztahu (3.4) byla vypočtena výběrová směrodatná odchylka aritmetického průměru změřeného zpoždění $\overline{s_{l_a}} \doteq 0{,}532\ \mu s$ (pro paket velikosti 512 B), což je, vzhledem k požadavkům na měření, dostatečná přesnost.

$$\overline{s_{l_a}} = \sqrt{\frac{\sum_{i=1}^{n}\left(l_i - \overline{l_a}\right)}{n(n-1)}} \tag{3.4}$$

Každé měření bylo zopakováno 10 ×, zpoždění uváděné v tab. 3.1 je jejich aritmetickým průměrem. Další měření platformy VNUML, viz kap. 4.1.2, tab. 4.1 a obr. 4.5. Během měření bylo ověřeno, že priorita protokolu ICMP je v systému VNUML, resp. v rámci „UML switch", nastavena stejně jako priorita ostatního provozu.



Obr. 3.1: Srovnání zpoždění

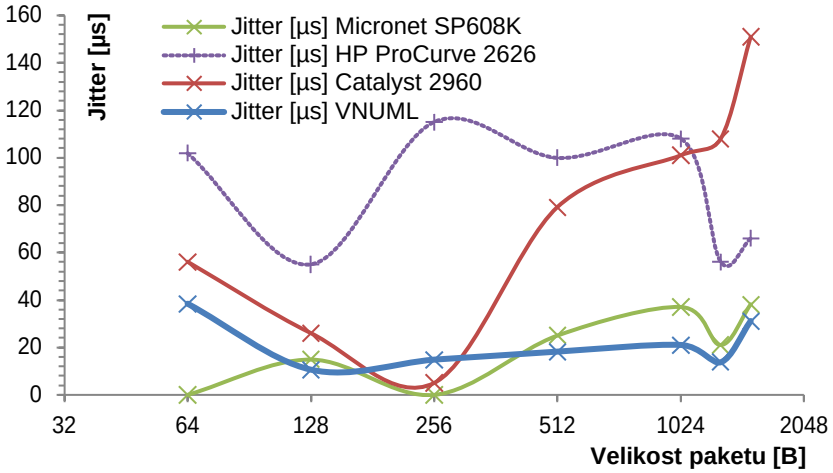
V souladu s předpokladem bylo změřeno nejmenší zpoždění na přepínači Catalyst. Jak je patrné z obr. 3.1, „UML switch“ vkládá v rozsahu 64–1 024 B přibližně konstantní zpoždění. Konstantní zpoždění je neočekávané při zohlednění použité metody měření. Program Ping není pro podobná měření navržen a je možné, že jeho přesnost ovlivnila měření. Zmíněný tester VeEX je naopak přesný profesionální analyzátor.

3.1.2 *Test jitteru*

Testy jitteru jsou založeny na stejné metodologii jako testy zpoždění (viz kap. 3.1.1). Udávaný jitter je vypočten podle metodiky, viz kap. 3.1 a je opět aritmetickým průměrem výsledků opakovaných měření provedených se stejnou velikostí paketu. Nejmenší proměnlivost zpoždění vnášel přepínač Micronet, UML switch v rozsahu 128–1 024 B dosahoval rovněž stabilních zpoždění, resp. malého jitteru. Všechny získané výsledky jsou shrnuty v tab. 3.2 a v grafu na obr. 3.2.

Velikost paketu [B]	**Jitter [µs]**			
	VNUML	**Catalyst 2960**	**Micronet SP608K**	**HP ProCurve 2626**
64	38,33	56,00	0,00	102,00
128	10,56	26,00	15,00	55,00
256	14,67	5,00	0,00	115,00
512	18,11	79,00	25,00	100,00
1024	20,89	101,00	37,00	108,00
1280	13,78	108,00	21,00	56,00
1518	31,00	151,00	38,00	66,00

Tab. 3.2: Srovnání jitteru



Obr. 3.2: Srovnání Jitteru

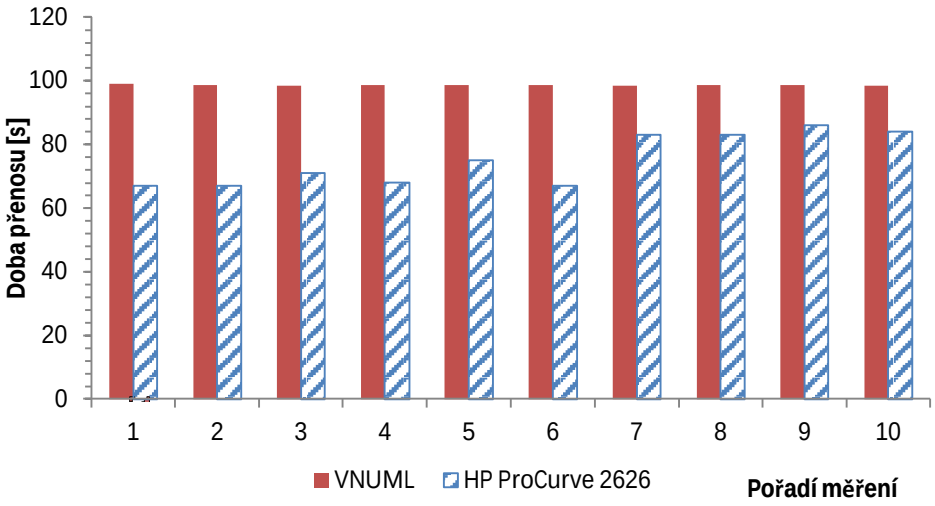
3.1.3 *TEST PŘENOSU SOUBORŮ*

Předposledním měřením dostupných síťových prvků, které bylo provedeno, bylo měření přenosu souborů. Vybavení pracoviště pro testování a testované přepínače byly stejné, jako v případě testu zpoždění, viz kap. 3.1.1. Pro testování přenosu souborů byl použit protokol FTP. Test byl zaměřen zejména na ověření propustnosti přepínače UML switch, viz kap. 4.1. Byl přenášen soubor o velikosti 236 254 243 B přes síť z jednoho virtuálního počítače do druhého.

Jako referenční přepínač byl pro toto měření zvolen přepínač HP ProCurve 2626. Přenos souborů přes přepínač ProCurve byl měřen s použitím 2 PC (Intel Core 2 CPU 630 @ 1.86 GHz 1.87 GHz; 2GB RAM; HDD 148 GB, WDC WD1600JS-00NCB1 ATA). Rychlost přenosu souboru byla měřena 10× a dle vztahu (3.2) průměrována. Výsledky jsou shrnuty v tab. 3.3 a na obr. 3.3.

	Přenos VNUML [min]	**Přenos HP ProCurve 2626 [min]**
Minimální čas	1:38 (98,405 s)	1:07 (67 s)
Maximální čas	1:38 (98,875 s)	1:26 (86 s)
Průměrný čas	1:38 (98,524 s)	1:15 (75 s)

Tab. 3.3: Přenos 236MB souboru



Obr. 3.3: Přenos 236MB souboru, 10 měření

Na obr. 3.3 je možné pozorovat vysokou stabilitu při přenosech souborů přes UML switch a naopak nízkou stabilitu přepínače HP ProCurve. Nižší stabilita HP může být zapříčiněna provozem s použitím reálných PC v reálném prostředí.

3.1.4 TESTY ZTRÁTOVOSTI RÁMCŮ

Test ztrátovosti rámců (Frame Loss Test) je součástí doporučení RFC 2544, tento test byl proveden na všech testovaných zařízeních. Pro srovnání výsledků byl navíc do testu přidán zastaralý HUB 3Com Corporation – Superstack II. Primárním důvodem pro zařazení HUBu do testu byly velmi dobré výsledky všech testovaných přepínačů. HUB sloužil primárně pro ověření funkčnosti použité metodiky.

Test ztrátovosti rámců je popsán v doporučení RFC 2544 [97] v kapitole 26.3. Ztrátovost rámců P_l je definována jako počet rámců, které by měly být přeneseny, ale nejsou [97]. V RFC 2544 je ztrátovost rámců definována rovnicí (3.5), kde počet vstupních rámců je označen c_{pi} a počet výstupních c_{po}.

$$P_l = \frac{(c_{pi} - c_{po})}{c_{pi}} \cdot 100 \tag{3.5}$$

Na počátku testování byly rámce vybrané velikosti zasílány maximální přenosovou rychlostí. Rychlost zasílání byla postupně snižována – vždy o 10 % z předchozího měření. Postup byl opakován, dokud nebyla změřená ztráta 0 %. Výsledky tohoto testu jsou vyneseny v tab. 3.4, která udává rychlost, při které měřené aktivní prvky dosáhly nulové ztrátovosti při dané velikosti paketu. Výsledky měření VNUML a přepínače Micronet jsou sloučeny do jednoho sloupce, protože výsledky jsou zcela shodné. Analogicky byly sloučeny výsledky měření Catalystu a přepínače HP ProCurve.

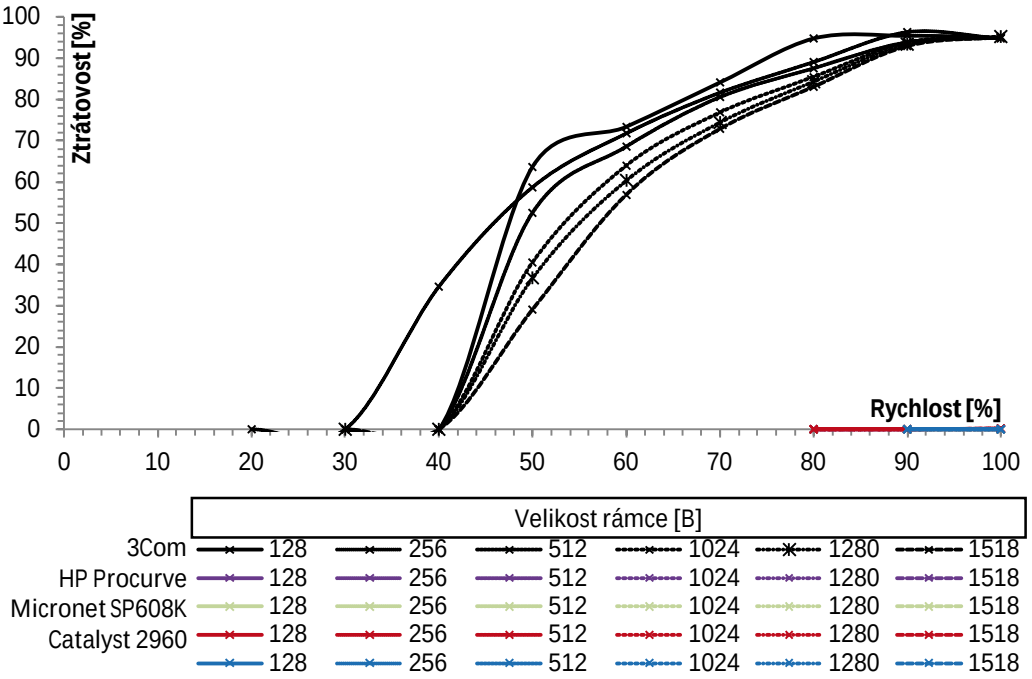
UML switch (VNUML) a Micronet měly ztrátovost 0 % pro všechny testované velikosti paketů se 100% přenosovou rychlostí (tedy linkovou rychlostí), což je výborný výsledek. Přepínače Catalyst a ProCurve měly shodně ztrátovost 0 % až při 90 % linkové rychlosti.

HUB 3Com dosahoval v souladu s předpoklady velmi špatných výsledků, ale jeho zahrnutí do testovací množiny bylo pro srovnání kvality přepínačů nezbytné. Nízká přenosová rychlost s 0% ztrátovostí je dána kolizemi paketů, jejichž příčinou je princip funkce HUBu.

Velikost paketu [B]	Rychlost [%]		
	VNUML + Micro-net SP608K	Catalyst 2960 + HP ProCurve 2626	3Com Superstack II
128	100	90	40
256	100	90	30
512	100	90	40
1024	100	90	40
1280	100	90	40
1518	100	90	40

Tab. 3.4: Souhrn měření ztrátovosti (rychlost udávána v % linkové rychlosti)

Kompletní výsledky měření ztrátovosti jsou graficky reprezentovány na obr. 3.4, změřené hodnoty jsou uvedeny v tabulce, viz příloha B.1. Nápadný je především graf závislosti ztrátovosti na přenosové rychlosti HUBu 3Com – průběh je vynesen černou barvou. Průběhy všech ostatních přepínačů jsou promítnuty na ose x.



Obr. 3.4: Souhrn měření ztrátovosti (rychlost udávána v % linkové rychlosti)

3.1.5 *ZHODNOCENÍ MĚŘENÍ*

Provedená měření poskytla několik překvapivých výsledků. Největším překvapením bylo zpoždění a stabilita přepínače UML switch, která byla lepší než v případě přepínače HP ProCurve 2626. Zajímavé výsledky byly dosaženy rovněž při testování přenosu souborů.

Před provedením testu ztrátovosti nebylo předpokládáno, že Cisco Catalyst dosáhne 0% ztrátovosti až při 90 % linkové rychlosti a ne při 100 %.

Systém VNUML se zdá být na první pohled pro vývoj přepínačů použitelným nástrojem. Přesto bylo důsledkem podrobných měření jeho opuštění, viz kap. 4.1.3. Ačkoli testování proběhlo na velmi malé síti, všechny testované přepínače splnily samostatně podmínky sta-

novené v kap. 1.1. Měření posloužilo jak k prověření vlastností VNUML, tak ustanovilo reálné měřítko dalšímu výzkumu.

3.2 ARCHITEKTURA

Navrhovaný aktivní prvek – přepínač – je určen do role PE, tedy do přístupových bodů operátora. Do této role jsou určena např. zařízení Cisco Catalyst 4500, 6500, 6800, Juniper řady EX, případně Brocade řady VDX, v závislosti na velikosti operátora a pokrývané oblasti. Na PE zařízení (ať již L3 přepínače nebo směrovače), viz kap. 1.2, jsou kladeny značné požadavky – jak z hlediska klasifikace a značkování provozu, tak bezpečnosti, minimalizace zpoždění, tak také z hlediska stability, redundance, aktualizovatelnosti bez restartů apod. Obvykle se jedná buď o stohovatelná (stackovatelná) zařízení, nebo větší šasi vybavené sadou linkových karet a supervizorovou kartou.

Celkové počty portů v jednom šasi se mohou pohybovat v řádech stovek portů – např. sedmislotové šasi Catalyst 6807-XL lze osadit 5 linkovými kartami (2 sloty jsou určeny pro supervizorové karty), pokud jsou použity např. 1Gb linkové karty WS-X6748-GE-TX, jedná se o 240 portů. Linkovými switch kartami lze osadit i např. 16slotový systém CRS-3 (i když v tomto případě jsou zajímavé např. osazení kartami 40× 10Gb). Různé architektury těchto zařízení jsou stručně popsány v kap. 1.4 a 1.5, dále pak v literatuře [55] a např. [72]. Pokud nebudou uvažovány multicasty, je počet možných propojení v daném prvku dán 2člennou kombinací (na pořadí nezáleží) počtu portů, resp. vstupů/výstupů N, tedy vztahem (3.6). Pro zmíněný příklad Catalystu 6807-XL je $c_N = 28\,680$ možných kombinací. Pokud jsou uvažovány i multicast pakety, jedná se o 57 600 možností propojení fyzického portu na fyzický port.

$$c_N = \binom{N}{2} = \frac{N!}{(N-2)! \cdot 2!} \tag{3.6}$$

Z hlediska výkonnosti je třeba uvážit, že přepínač může být připojen do rozsáhlejší sítě. Na vyšších vrstvách RM ISO/OSI může být počet zpracovávaných adres mnohem vyšší. Při zpracování L2 RM ISO/OSI může hrát omezující roli kapacita CAM tabulek (Content addressable memory), ale ani při jejím překročení není prvek vyřazen z provozu, viz [55].

Při vyhledávání cílového uzlu přepínač rozlišuje směr rámce/paketu, rozlišuje tedy mezi vstupním a výstupním portem – v rámci L2 protokolu pracuje s rozdílnou adresou. Počet možných propojení je 2člennou variací a je dán rovnicí (3.7).

$$c_{Nv} = \frac{N!}{(N-2)!} \tag{3.7}$$

Výzkum přepínacích algoritmů je vhodnější zahájit prvkem s výrazně menším počtem možných vnitřních propojení, zejména kvůli možnosti identifikovat chyby. Jako ideální se jeví použití 3 nebo 4 portového přepínače. V případě 4portového prvku bude $c_N = 6$ a $c_{Nv} = 12$. Ve zvláštních případech, viz kap. 4.1, byl kvůli výkonnosti simulace snížen počet vstupů/výstupů přepínače na 3. Z hlediska analýzy funkčnosti se ale nejedná o vhodné řešení, model 3portového přepínače byl opuštěn, viz kap. 4.1.3.

3.2.1 STRUKTURA SPOJOVACÍHO POLE

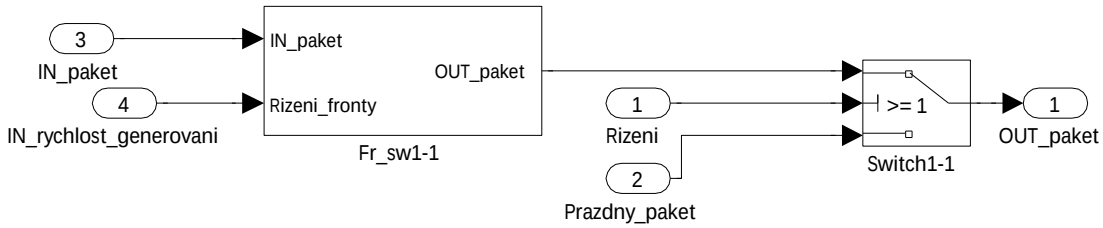
Některé vybrané struktury spojovacího pole jsou zmíněny v kap. 1.4 a 1.5.4. Topologie spojovacího pole může významným způsobem ovlivnit celkovou výkonnost aktivního prvku. Při

návrhu spojovacího pole je třeba zohlednit i problematiku vnitřního blokování, které může jeho výkonnost zásadně ovlivnit.

Pro potřeby zapojení neuronové sítě do řízení spojovacího pole se jeví jako výhodné použít modifikovanou strukturu křížového spínače. Křížový spínač v roli spojovacího pole se jeví jako výhodný proto, že umožňuje snadné zapojení ovládání externím algoritmem. Ačkoli lze křížový spínač navrhnout tak, aby se jednalo de facto o samosměrovací pole, z hlediska implementace požadované funkcionality je výhodnější centrální řízení prvku, se samostatným ovládáním každého z N^2 spínacích prvků, viz kap. 1.4.4. Okrajově byla zvažována i spojovací pole typu „full mash", tedy plně propojené porty. V případě 4portového modelu by se jednalo o výhodné zapojení bez problémů s vnitřním blokováním. Nárůst spojení s počtem portů je evidentní nevýhodou této topologie.

Spojovací pole lze doplnit o vyrovnávací paměti, které mohou pozitivně ovlivnit jeho propustnost. V případě křížového spínače lze paměti umístit na jeho vstup, výstup nebo do každého jednotlivého spojovacího bodu. Z hlediska vnitřního blokování je nejvýhodnější použití paměti přímo před každým spojovacím bodem. Křížový spínač doplnění vyrovnávacími pamětmi byl modelován – ukázka konstrukce modelu jednoho ze spojovacích bodů, viz obr. 3.5. Blok `Fr_sw1-1` na obr. 3.5 je modul vyrovnávacích pamětí – je popsán dále. Blok `Switch1-1` je jeden spojovací bod. Ostatní bloky reprezentují řízení, vstupy a výstupy ze zbytku modelu.

Obvykle byla nastavena velikost vyrovnávací fronty na 2 takty (rámce) modelovaného paketu. Velikost vyrovnávací fronty lze v modelu libovolně měnit, uvedená kapacita ale pokryla většinu kolizí. Větší zásobník při zaplnění působil zbytečné zpoždění – z pohledu QoS je efektivnější takové rámce zahodit.



Obr. 3.5: Model spojovacího bodu s předřazenou vyrovnávací pamětí

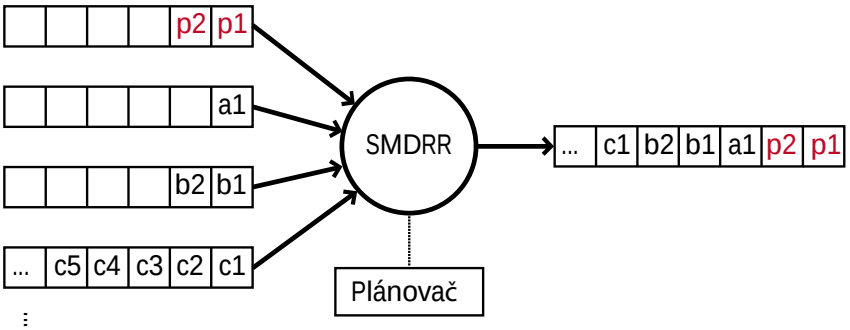
Vytvořený model spínacího bodu byl využíván jen výjimečně – jeho použití komplikovalo vyhodnocování výsledků simulací. Pozitivní vliv na QoS přitom nebyl, vzhledem k ostatním modelovaným parametrům, zásadní.

Zvolené řešení s odděleným řízením, resp. oddělenou přepínací (řídicí, konstelační) maticí a křížovým spínačem, je z hlediska modelování výhodné – řízení křížového spínače lze snadno měnit a také vytvářet paralelně pracující struktury umožňující porovnávat výsledky.

3.2.2 SOFTWAROVÉ FRONTY A PLÁNOVÁNÍ

Další konstrukční částí s přímým vlivem na výkon přepínače a zejména na QoS je návrh implementace a správa softwarových front. Stručný popis problematiky, viz kap. 1.3, zejména pak 1.3.2 až 1.3.4.

Z popsaných algoritmů byly uvažovány pro implementaci do modelů zejména algoritmy LLQ a MDRR. LLQ, jako v současné době nejpoužívanější algoritmus, MDRR je zajímavý svojí jednoduchostí a rychlostí. Pro účely řešené problematiky byl algoritmus MDRR (obr. 1.11) dále upraven – do režimu striktní priority pro všechny fronty, viz obr. 3.6.



Obr. 3.6: Upravený algoritmus MDRR – striktní priorita

Softwarové fronty náležící upravenému algoritmu MDRR, dále jen SMDRR, jsou plánovačem čteny postupně od fronty s nejvyšší prioritou, která je přečtena celá, následuje přečtení fronty s nižší prioritou atd. Je zřejmé, že čím má fronta v rámci SMDRR nižší prioritu, tím více strádá. Výhodou SMDRR je ale velmi transparentní provoz, který umožňuje snadno identifikovat chyby v dalších částech modelů. Kromě SMDRR byl modelován i konveční alternující MDRR se striktní prioritou nejdůležitější fronty, v tomto případě ke strádání front s nižší prioritou nedochází, viz kap. 1.3.3.

Zahazování paketů bylo možné řešit několika metodami, viz kap. 1.3.4. Pro modelování se nejlépe hodí jednoduchý mechanismus „Tail Drop“, který je spolu s odvozenými mechanismy v současné době široce používán i v komerční praxi. Postupně byly navržené modely upraveny tak, aby byla modifikovatelná rychlost generování paketů zpětnou vazbou, je tedy možné generovat i pokročilé metody, jako je RED a WRED. Za dominantní, zejména při komplexním návrhu, byla ale zvolena, zejména kvůli transparentnosti, metoda „Tail Drop“ – pakety, které by byly zařazeny do fronty, ale přesáhly by její velikost navržený přepínač, resp. model, zahodí.

Při komplexním návrhu přepínače byly, vyjma umístění přímo před jednotlivé spínací body, viz kap. 3.2.1, zvažovány i zbývající možnosti umístění vyrovnávacích pamětí, resp. softwarových front vzhledem ke křížovému spínači – tedy před a za něj. Nejlépe se osvědčila instalace softwarových front před křížový přepínač. Tato varianta je použita i v komplexním modelu. Toto umístění se z hlediska teoretických poznatků jeví nejméně výhodné. Dochází v rámci něj k blokování rámců – ve vstupních frontách jsou zařazeny pakety, které by mohly projít spojovacím polem, ale protože i fronty s podporou QoS mají pro rámce stejné priority charakter FIFO, nemohou být odeslány. Řešení SW front před spojovacím polem ale řeší součástně několik problémů:

- Obdobně jako umístění SW fronty za spojovací pole řeší problém plánování a QoS.
- Současně řeší problém kolizí ve spojovacím poli (soutěž o cestu spojovacím polem).
- Spojovací pole nemusí být tak výkonné – v jednom taktu může projít spojovacím polem pouze rámec potřebný pro odeslání – není nutné zpracovat veškerý příchozí provoz.
- Na výstupu spojovacího pole nejsou potřeba žádné fronty. Softwarové fronty před spojovacím polem může plánovač vyprazdňovat takovou rychlostí, která odpovídá požadavkům výstupních rozhraní. Přes tuto možnost se při modelování ukázalo být efektivnější doplnění o FIFO fronty na každém výstupním rozhraní, viz dále.

Výstupní („hardwarové“) FIFO fronty byly ve všech případech umístěny konvenčně na výstupní rozhraní.

3.3 Aplikace neuronových sítí na řízení aktivního prvku

Umělou neuronovu síť lze v rámci přepínače, resp. jeho řízení, využít několika způsoby. Využití neuronových sítí se nabízí jako jedno z možných řešení problémů, které musí aktivní prvek řešit. Neuronové sítě vynikají v úlohách, jejichž řešení lze popsat deterministickými algoritmy jen obtížně nebo vůbec. V obecné rovině se problematikou neuronových sítí zabývá literatura např. [101] a základy také např. [102], dále viz kap. 1.6.2. Postupně byly vybrány následující problémy přepínačů, k jejichž řešení lze neuronovu síť využít.

Klasifikace provozu je nutným krokem před jakýmkoli dalším zpracováním provozu v rámci QoS. Neuronovu síť lze využít k rozpoznávání typů provozu, protokolů a jejich zařazování do toků. Neuronovu sítí lze, v teoretické rovině, nahradit např. NBAR v Cisco prvcích, viz kap. 1.2.5. Výhodou takového řešení by byla zejména, možnost zařazení do třídy pro zařízení neznámých protokolů na základě jejich charakteristik a podobnosti s obdobnými protokoly. Proběhlo několik experimentů s cílem neuronovou síť pro identifikaci typu provozu použít, viz kap. 4.2.

Správa front a zejména plánovač může být nahrazen neuronovu sítí. Problém použití neuronové sítě v roli plánovače je její trénování tak, aby byla správa front efektivnější než konvenčním algoritmem. Konvenční algoritmy mají vyčíslitelnou efektivitu pro různé typy provozu, plánovač pracuje s jasně definovanou množinou toků (značek, resp. typů provozu). Neuronová síť v roli plánovače, pokud není přetrénovaná, nabízí oproti deterministickým algoritmům pouze vyšší míru nejistoty.

Řízení spojovacího pole je obvykle buď samosměrovací nebo je řízeno deterministickým algoritmem, viz kap. 1.4, 1.5.4 a 3.2.1. Neuronová síť může buď spojovací pole nahradit – rámce/pakety mohou procházet neuronovou sítí, přičemž vstupním vektorem neuronové sítě je provoz přicházející ze vstupních portů a výstupní vektor by měl odpovídat provozu ukládanému do HW FIFO front. Druhou možnou aplikací neuronové sítě k řízení spojovacího pole je použít konvenční spojovací pole – např. křížový spínač a neuronovou sítí řídit jeho činnost. Neuronová síť zajistí vytvoření správné cesty ve spojovacím poli.

Z vyjmenovaných 3 možných se výzkum nejvíce soustředil na řízení spojovacího pole. Jako velmi perspektivní se také jevilo využití neuronových sítí v klasifikaci provozu. Bylo třeba zvolit pro jednotlivé aplikace vhodnou neuronovu síť.

3.3.1 Volba Neuronové sítě

Volba vhodné neuronové sítě použité pro řízení spojovacího pole byla provedena na základě publikací [85], [86], [87] a zejména [90]. Pro řešení úloh obsahujících problematiku rozpoznávání schémat (pattern recognition) doporučuje literatura [90] použití algoritmus zpětného šíření chyby.

Pozn. Technická realizace neuronových sítí je mj. prováděna obvykle prostřednictvím tzv. neuročipů (neuronových čipů). Vývojem neuročipů se t. č. (r. 2014) intenzivně zabývá IBM Research [103].

V úvodní fázi projektu byly rovněž ověřovány možnosti použití jiných, jednodušších typů neuronových sítí, např. perceptronová síť. Protože řízení spojovacího pole může být de facto binární, zdá se být pro modelování takového systému perceptronová síť ideální. Výzkum prokázal, že perceptronovou síť lze použít pro řízení spojovacího pole, viz kap. 3.3.3, pro úlohy spojené s klasifikací provozu se nehodí.

Pozn. Perceptron je neuronová síť, která má jen jeden pracovní neuron (neuron ve skryté vrstvě). Vstupní signál perceptronu může být jen binární nebo bipolární signál. Počet neuronů ve vstupní vrstvě není omezen. Pracovní neuron je spojen se všemi vstupními neurony. Spojení mezi vstupním a pracovním neuronem mají nastavena různou váhu [102].

Vzhledem k výsledkům výzkumů popsaných v [85], [86], [87] a v doporučeních uvedených v [90], byla pozornost zaměřena zejména na sítě založené na algoritmu zpětného šíření chyby (feedforward backpropagation).

3.3.2 Algoritmus zpětného šíření chyby

Backpropagation je adaptační algoritmus patřící mezi metody učení s učitelem. Česky je označen jako algoritmus zpětného šíření chyby. Je to nejčastěji používaný algoritmus učení vícevrstvých neuronových sítí. Podle [102] je tento algoritmus použit v přibližně 80 % všech použití neuronových sítí. Pokud mají neurony sigmoidní aktivační funkci, je vícevrstvá architektura sítě tou nejobvyklejší při jejich propojování.

Pozn. V při využívání neuronových sítí se k jejich učení, resp. tréninku, používají nejčastěji 3 typy učení. Jedná se o:

Trénink s učitelem (supervised learning) – pracuje se s tréninkovou množinou (vstupní a výstupní matice – tréninkový vzor). Při tréninku s učitelem je v rámci tréninkové množiny znám vstupní i očekávaný výstupní vektor. Během procesu učení se v rámci neuronové sítě v každém kroku nastavují váhy tak, aby výstupy (výstupní vektor) odpovídal co nejvíce ověřovací množině (resp. očekávanému výstupnímu vektoru).

Klasifikované učení – učitel hodnotí kvalitu výstupu sítě pro daný vstup pomocí známky.

Samoorganizaci (učení bez učitele) – tréninková množina je dána pouze vstupy sítě. Síť odhaduje souborné vlastnosti tréninkových vzorů.

Pozn. **Aktivační funkce** je matematická transformace souhrnu vah vstupů tvořící výstup neuronu. Aktivační funkce má dvě části [90], [102].

První část je **kombinační funkce** (combination function), která sloučí všechny vstupy neuronu do jedné hodnoty. Kombinačních funkcí je několik. Nejznámější kombinační funkcí je vážený součet. Při použití této metody je každý vstup neuronu násoben svojí váhou [90], [102].

Druhá část aktivační funkce se nazývá **přenosová funkce** (transfer function), jejímž vstupem je výstup kombinační funkce. Přenosová funkce provádí nad vstupy obvykle nelineární transformaci a generuje výstup neuronu. Typ přenosové funkce je odvislý od typu kombinační funkce, kterou neuron obsahuje [104]. Aktivační funkce vyjadřuje „míru rozhodnosti“ každého z neuronů v síti. Aktivační funkci lze měnit, např. ji lze přizpůsobit tréninkové množině. Přizpůsobení aktivační funkce ale zpomaluje trénování sítě, protože je třeba v síti měnit další parametr. Standardní aktivační funkce je sigmoidní. Sigmoidními aktivačními funkcemi jsou např. **lineární** (linear, v Matlabu funkce `purelin`); **ostrá nelinearita**; **saturovaná lineární funkce**; **standardní**, logistická, sigmoida (log-sigmoid v Matlabu funkce `logsig`); **hyperbolický tangens** (tan-sigmoid, v Matlabu funkce `tansig`) [90], [102].

Podle průběhu aktivační funkce se mohou neuronové sítě rozdělovat dále na lineární, nelineární (většina systémů je nelineární) a prahové.

Podle průběhu aktivační funkce lze sítě dělit také na diskrétní a analogové podle toho, jestli je průběh aktivační funkce neuronu spojitý nebo diskrétní [102].

Feedforward (česky dopředné šíření vstupního signálu tréninkového vzoru) je první etapou ze třech při učení vícevrstvé sítě metodou backpropagation. Další etapy jsou zpětné šíření chyby (backpropagation) a aktualizace, resp. úprava vah spojů mezi neurony [102].

Algoritmus, resp. neuronová síť „feedforward backpropagation", je typ vícevrstvé neuronové sítě s jednou skrytou vrstvou [90].

3.3.3 KOMPARAČNÍ TEST NEURONOVÉ SÍTĚ

Všechny typy neuronových sítí, které byly v rámci výzkumu použity, byly podrobeny testům, jejichž úkolem bylo ověřit správnou funkci parametrů a proměnných, které lze při vytváření a trénování sítě, resp. jejího modelu, použít. Bylo třeba ověřit chování modelovacího prostředí, specifika implementace numerických integračních metod, nastavení simulačního algoritmu a zejména zvolit vhodnou neuronovou síť pro konkrétní aplikaci.

Nejjednodušším modelem, který byl vytvořen a použit pro otestování neuronových sítí, ověření základní funkčnosti simulačního softwaru a nastavitelnosti všech parametrů, byl model booleovské funkce antivalence [105]. Ta bývá někdy označována jako operace exkluzivní disjunkce, exkluzivní OR nebo, v technické praxi nejčastěji, XOR.

XOR je běžná logická operace, která se často používá v elektrotechnice, kryptografii, informatice atp. Její vlastnosti lze popsat např. pravdivostní tabulkou, viz tab. 3.5, matematická definice antivalence, viz [105].

P1	P2	T= P1ÅP2
1	1	0
1	0	1
0	1	1
0	0	0

Tab. 3.5: Pravdivostní tabulka logické funkce antivalence (XOR)

Logická operace XOR byla zvolena pro svoji jednoduchost a s tím spojenou snadnou implementovatelnost. V počátku byla idea modelování XOR inspirována také dostupností literatury, viz kap. 1.6.3, která model popisovala. Výsledky simulací jsou při simulování XOR ověřitelné na první pohled. Další výhodou použití XOR je malá časová náročnost tréninku vytvořené sítě. Např. neuronová síť typu feedforward backpropagation potřebuje ke korektnímu modelování operace XOR pouze dva neurony ve skryté (pracovní) vrstvě. Operace XOR byla zvolena také pro svůj samotný charakter – blízkost řešené problematice – v případě XOR, stejně jako v případě řízení přepínače, záleží na pořadí (např. na rozdíl od OR – logické funkce disjunkce, viz tab. 3.6).

P1	P2	T=P1∨P2
0	0	0
0	1	1
1	0	1
1	1	1

Tab. 3.6: Pravdivostní tabulka logické fuknce disjunkce (OR)

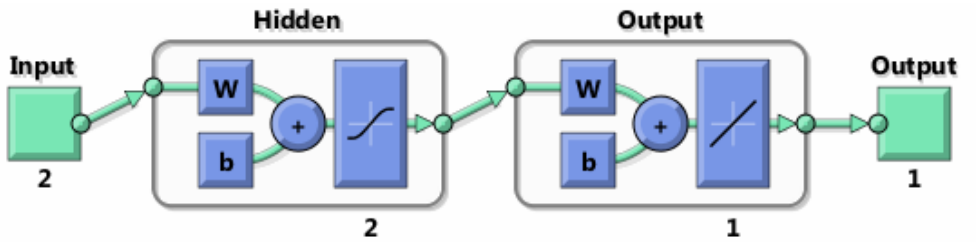
Tréninkovou množinu pro modelování XOR tvořily 2 matice odvozené z tab. 3.5. Vstupní matice **P**, viz (3.8) (input data), odpovídá sloupcům P1 a P2 tab. 3.5 a cílová matice **T**, viz (3.9) (target data), odpovídá sloupci T tab. 3.5.

$$\mathbf{P} = \begin{pmatrix} 0 & 0 & 1 & 1 \\ 0 & 1 & 0 & 1 \end{pmatrix} \tag{3.8}$$

$$\mathbf{T} = \begin{pmatrix} 0 & 1 & 1 & 0 \end{pmatrix} \tag{3.9}$$

V prvních fázích výzkumu byl k návrhu a tréninku neuronové sítě krátkodobě používán nástroj nntool – součást knihoven „Neural Networks Toolbox“ (MATLAB). Neuronové sítě vytvořené tímto nástrojem nedosahovaly požadovaných výsledků. Identifikované problémy a výsledky dosažené s použitím tohoto nástroje jsou stručně popsány v příloze C.2. Po ukončení používání nástroje nntool byly všechny další neuronové sítě vytvářeny výhradně konzolovými nástroji, resp. skripty, viz závěr přílohy C.2.

Testování bylo zahájeno návrhem a modelováním XOR neuronovou sítí typu feedforward backpropagation. Tréninková funkce (training function) byla vybrána TrainLM (Levenberg – Marquardt, viz dále), adaptační funkce byla zvolena s klesajícím gradientem (gradient descent), dynamickými vahami a biasem (momentum weight and bias) – LearnGDM [90]. Výkonová funkce – MSE, viz dále. Síť byla vytvořena se 2 vrstvami, přičemž pracovní vrstva (layer 1) obsahovala 2 neurony. Přenosová funkce (transfer function) byla zvolena sigmoidní „hyperbolický tangens“ (tan-sigmoid), v Matlabu funkce `tansig`. Síť je schematicky znázorněna na obr. 3.7.



Obr. 3.7: Schematické znázornění modelu XOR

Algoritmus učení Levenberg – Marquardt (TrainLM): Trénování, resp. učení sítě metodou TrainLM, tedy Levenberg – Marquardt, je pro většinu aplikací nejrychlejší backpropagation algoritmus z dostupných v Neural Network toolboxu. Parametry, které je možné nastavit při použití metody TrainLM, jsou uvedeny v příloze C.3.

Pro dosažení žádaných výsledků je třeba nastavit parametry učení sítě tak, aby pro trénování byly využity všechny prvky z tréninkové množiny. V případě trénování XOR má vstupní matice (3.8) čtyři sloupce a dva řádky. Pokud nejsou použity pro trénování všechny prvky, dosažené výsledky neodpovídají očekávání.

Nastavení poměru tréninkové a ověřovací množiny zajišťují parametry funkce rozdělující množinu pro trénování – `trainRatio`, `valRatio` a `testRatio`. Pokud je `sitf` označení pro vytvořenou neuronovou síť, je podoba parametrů následující:

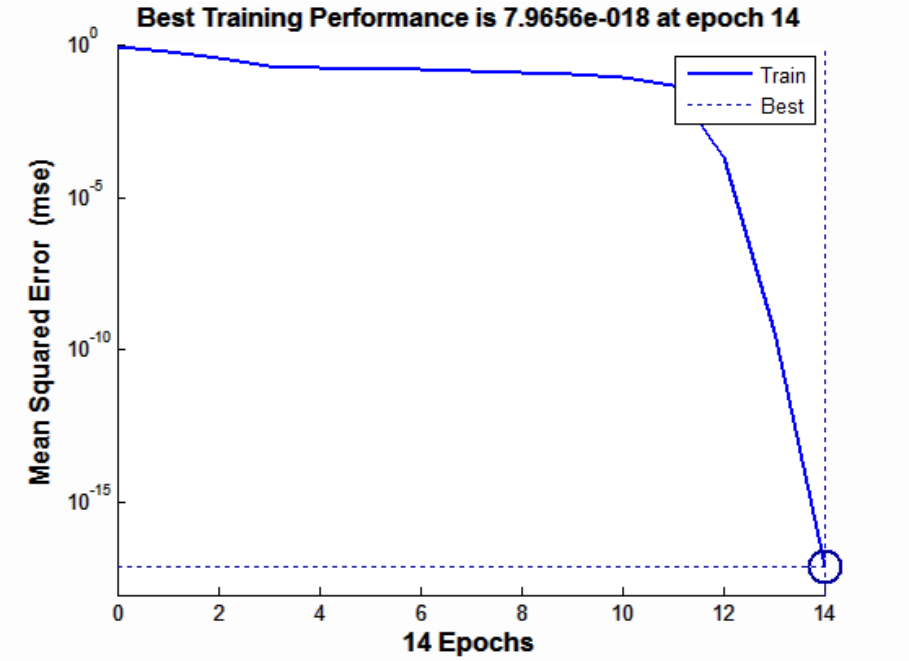
```
sitf.divideParam.trainRatio = 90/100;
sitf.divideParam.valRatio = 5/100;
sitf.divideParam.testRatio = 5/100;
```

Pro trénování je použito 90 % dat vstupní matice, pro ověření a testování pak 5 %. Funkce `divideParam` je popsána v [90].

Zjištěné poznatky byly zohledněny při návrhu skriptu, který vytvoří a natrénuje síť s žádanými a očekávanými výsledky (skript byl pro přehlednost zkrácen o parametry, které jsou s výchozím nastavením MATLABu 2010b shodné):

```
p = [0 0 1 1;
     0 1 0 1];
t = [0 1 1 0];

hiddenLayerSize = 2;
sitf = fitnet(hiddenLayerSize);

sitf.divideParam.trainRatio = 90/100;
sitf.divideParam.valRatio = 5/100;
sitf.divideParam.testRatio = 5/100;

[sitf,tr] = train(sitf,p,t);

outputs = sitf(p);
errors = gsubtract(t,outputs);
performance = perform(sitf,t,outputs)

sitf (p)

view(sitf)
```

S použitím navrženého skriptu byly dosaženy výsledky odpovídající očekávání. Síť dosáhne cílového stavu po přibližně 14 tréninkových epochách. Viz graf na obr. 3.8. Dosažený výkon (resp. chybovost mse) sítě byl $mse = 7{,}9656 \cdot 10^{-18}$.



Obr. 3.8: Trénování XOR pomocí skriptu – vývoj mse v průběhu trénování sítě

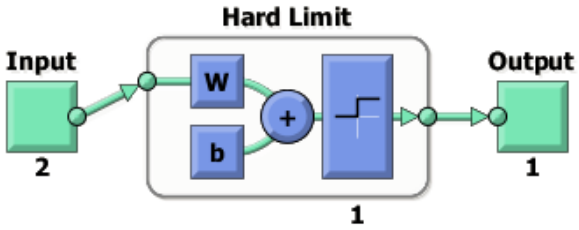
Střední kvadratická odchylka, označovaná v [90] jako **mse** (Mean Squared Error) je definována vztahem (3.10). Ve vztahu (3.10) označuje t_i množinu cílových hodnot při trénování neuronové sítě. Množina a_i je množinou skutečných výstupů.

$$F = mse = \frac{1}{N}\sum_{i=1}^{N}(e_i)^2 = \frac{1}{N}\sum_{i=1}^{N}(t_i - a_i)^2 \tag{3.10}$$

Podobně jako možnosti nastavení a vlastnosti sítě feedforward backpropagation byly zkoumány i vlastnosti perceptronu a sítě s rozpoznáváním vzoru (pattern recognition), kterou literatura [90] doporučuje jako náhradu perceptronu.

Perceptron je možné použít k vytvoření sítě modelující chování logické disjunkce (OR, pravdivostní tabulka viz tab. 3.6), ale ne pro modelování antivalence (XOR), pro což jsou třeba alespoň 2 neurony (perceptron je síť s pouze jedním neuronem [102]).

Schéma testovací sítě pro modelování OR, viz obr. 3.9. Síť byla tvořena dvěma neurony ve vstupní vrstvě. Síť byla natrénována během 3 iterací.



Obr. 3.9: Schematické znázornění modelu OR – perceptron

V případě sítě pattern recognition (v Matlabu také jako `patternnet`) bylo dosaženo výsledků horších, ale akceptovatelných, jako v případě sítě feedforward backpropagation. při stejné konfiguraci bylo dosaženo $mse = 1{,}1166 \cdot 10^{-7}$. Síť pattern recognition patří rovněž mezi sítě s dopředným šířením vstupního signálu tréninkového vzoru [90].

Provedené testy prokázaly, že pro řízení aktivního prvku se nejlépe hodí síť typu feedforward backpropagation, která bude v komplexním modelu použita přednostně.

4 NÁVRH PŘEPÍNAČE

Při naplňování cílů výzkumu vzniklo několik dílčích a dva komplexní modely síťového aktivního prvku přepínače. Jako první vznikly modely postavené nad systémem VNUML, průběžně pak komplexní modely vytvořené v MATLABu.

4.1 SIMULAČNÍ MODEL VNUML

Výzkum přepínačů a souvisejících algoritmů se kromě čistého modelování částečně orientoval také na uplatnění navržených algoritmů v rámci reálných softwarových přepínačů. Softwarové přepínače jsou samozřejmou částí systémů virtualizace hardwaru. Známými nástroji jsou např. Microsoft Hyper-V, produkty VMware (ESXi,…), Citrix XenServer, UML atd. Zmíněné systémy emulují chování reálného hardwaru, na kterém jsou provozovány různé operační systémy. Součástí virtualizace hardwaru je i virtualizace síťové infrastruktury, po které komunikují virtualizované servery a pracovní stanice.

Virtualizovaná infrastruktura bývá s fyzickou infrastrukturou propojena síťovým rozhraním serveru, na kterém je provozována. Tento typ propojení ale obvykle výkonem nedostačuje. Je známo několik řešení, jak přepínaní paketů ve virtuálním prostředí akcelerovat.

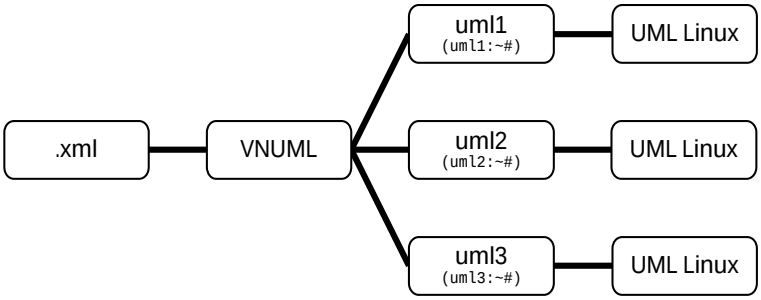
- § Umístit do fyzického serveru, nad kterým jsou provozovány virtuální servery, fyzickou rozšiřující kartu, která virtuálním serverům poskytuje adekvátní počet síťových rozhraní. V tomto řešení probíhá přepínání na rozšiřující kartě. Ta se ve fyzickém serveru chová jako několik fyzických síťových karet, přičemž každá je správou přidělena příslušnému virtuálnímu počítači/serveru. Řešení dodává např. firma Brocade, viz kap. 1.5.
- § Přepínání je realizováno na fyzickém přepínači s podporou virtualizace, kde probíhá přepínání. Ve virtuálním prostředí je pouze přítomen softwarový modul optimalizující komunikaci s fyzickým přepínačem.
- § Striktně softwarové řešení, v jehož rámci probíhá přepínání. Taková řešení jsou v současné době pravděpodobně nejčastější.

Během výzkumu byly části výsledků ověřovány zejména na open source řešení VNUML, resp. UML, kde byl nahrazen původní přepínací program a byly prověřovány možnosti jeho vylepšení. Jako nejperspektivnější řešení se jeví instalace fyzické akcelerační karty obsahující přepínací jádro a jeho poskytnutí virtuálním počítačům.

Pozn. Virtualizační systémy, jako např. VMware, emulují celý počítač, resp. server. Emulace zahrnuje veškeré hardwarové součásti včetně BIOSu. Taková virtualizace umožňuje kompletní izolaci systému, ale snižuje výkon celku a omezuje interoperabilitu hostovaného a hostitelského systému. Oproti tomu UML využívá odlišnou filosofii. Jádra hostovaných (virtualizovaných) systémů jsou spouštěna jako standardní neprivilegované procesy v uživatelském prostoru v rámci systému hostitelského. Jedná se de facto o tzv. aplikační virtualizaci. UML může být provozováno jen na hostitelských systémech od jádra verze 2.2.15. Kromě upraveného jádra je pro běh UML systémů potřeba vytvořit obraz souborového systému, ze kterého UML stroje bootují [106].

Pozn. Na španělské univerzitě Universidad Politécnica de Madrid (UPM) vznikl projekt Virtual Network User Mode Linux – VNUML, jehož vývoj skončil v r. 2009 [107]. Na projekt navázal projekt Virtual Networks over linuX (VNX) [108], jehož vývoj t. č. probíhá. Předkládaný výzkum proběhl v prostředí VNUML. VNUML je simulačním prostředím

pro simulování malých počítačových sítí. VNUML je postaven nad User Mode Linux (UML). VNUML umožňuje flexibilně pomocí Perl a XML skriptů vytvářet a spouštět virtuální sítě s de facto libovolným počtem uzlů. VNUML zásadně zjednodušuje využití UML jako nástroje pro modelování sítí. Princip spouštění simulace zachycuje obr. 4.1.

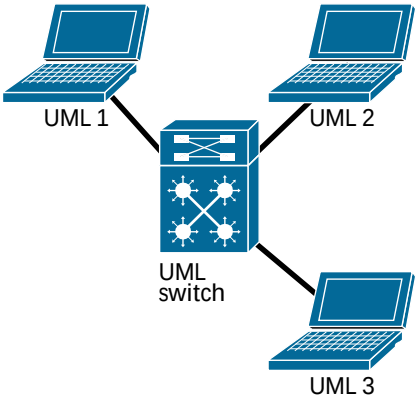


Obr. 4.1: VNUML model

Výhodou modelů postavených nad VNUML je skutečnost, že v rámci virtualizovaných linuxových uzlů lze provozovat libovolné aplikace a tedy i protokoly. Lze tedy pracovat např. se skutečnou implementací IP a dalších protokolů. Využitím, resp. úpravou, některých součástí UML systému, např. programu `uml_switch`, lze zásadně zrychlit vývoj vlastního modelu.

4.1.1 *Topologie*

Obr. 4.2 představuje základní topologii použitou pro vývoj a testování v rámci modelu vytvořeného ve VNUML, konfigurační soubor, viz příloha D.1. Vzhledem k relativně dlouhému spouštění celé vytvořené sítě bylo snahou minimalizovat počet zapojených uzlů. Současně bylo ale pro ověření správné funkce třeba zapojit více než 2 uzly. Uzly UML 1–3 na obr. 4.2 označují virtuální linuxové systémy. Uzel „UML switch“ označuje centrální prvek (program `uml_switch`), který byl v rámci výzkumu modifikován. Modifikovaný „UML switch“ umožňuje veškerý procházející provoz zrcadlit, resp. ukládat do souboru. I v průběhu simulace lze zaměňovat jednotlivé upravené centrální řídicí prvky.



Obr. 4.2: Model sítě se třemi koncovými uzly a jedním centrálním prvkem

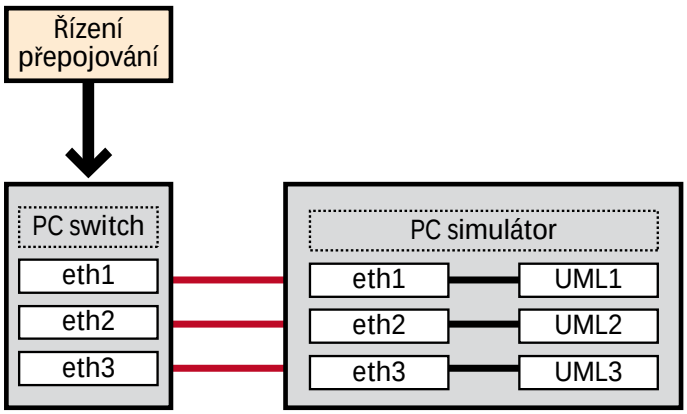
4.1.2 *Realizace*

Platforma VNUML byla zvolena kromě rychlé upravitelnosti také proto, že dávala naději na možné snadné převedení vytvořených modelů, resp. modifikací programu „UML switch“ do reálného provozu. Proto byl navržen a následně i sestaven emulátor složený ze 2 PC, viz obr. 4.3. Každé z PC bylo vybaveno 4 síťovými kartami, pro samotnou emulaci byla využita rozhraní eth1–3. Rozhraní eth0 bylo ponecháno pro kontrolní účely.

První z použitých PC, na obr. 4.3 označené „PC switch“, bylo využíváno jako emulátor přepínače. Byl na něm provozován upravený program „UML switch“, přičemž jeho vstupy/výstupy byly mapovány na fyzická rozhraní počítače eth1–3.

Druhé z použitých PC, na obr. 4.3 označené „PC simulátor“, sloužilo jako protistrana – umožňovalo emulaci 3 dalších počítačů – v tomto případě byly provozovány v rámci VNUML. Původním předpokladem bylo další rozšíření v rámci „PC simulátoru“, tedy simulace rozsáhlejší sítě uvnitř VNUML s tím, že rozhraní eth1–3 by byla použita jako trunk porty.

Celá snaha o fyzickou realizaci byla motivována možností jak testování, tak také provádět reálná měření. Ta probíhala v bodech, resp. propojeních vyznačených na obr. 4.3 a obr. 4.6 červenou čarou.



Obr. 4.3: Emulátor přepínače

Konfigurace zachycená na obr. 4.3 se neosvědčila – prokázal se příliš velký vliv topologie, resp. počet virtuálních uzlů v rámci „PC simulátoru“ na zpoždění, viz graf na obr. 4.5, tabulka naměřených hodnot viz tab. 4.1.

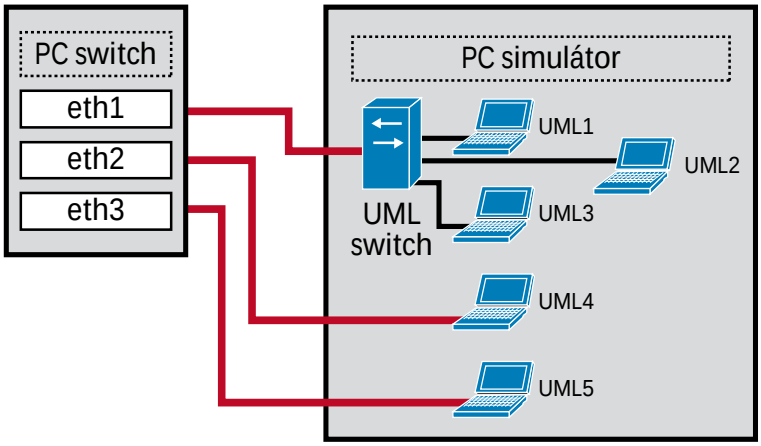
Konfigurace 1 uvedená v tab. 4.1 svojí topologií odpovídá konfiguraci uvedené na obr. 4.3. Každému rozhraní v rámci systému „PC switch“ odpovídá rozhraní v rámci „PC simulátor“. Měření zpoždění této konfigurace byla provedena pouze programem Ping tak, jak je uvedeno v kap. 3.1.1.

Velikost paketu [B]	**Zpoždění [µs]**			
	Analyzátor VeEX + 1 PC	**3 PC**	**Konfigurace 1**	**Konfigurace 2**
64	98,30	129,90	263,20	312,20
128	95,00	132,10	258,90	310,30
256	100,20	131,80	261,20	312,30
512	101,10	130,70	263,40	311,80
1024	99,50	126,80	264,60	313,50
1280	105,60	131,30	271,70	322,40
1518	108,40	164,60	265,00	325,20

Tab. 4.1: Vliv topologie na zpoždění

Konfigurace 2 uvedená v tab. 4.1 je zachycena na obr. 4.4. V této konfiguraci byla v rámci „PC simulátoru“ vytvořena virtuální síť postavená nad VNUML. Virtuální síť byla složena ze 3 uzlů označených na obr. 4.4 UML1–3. Jeden ze 4 portů virtuálního přepínače „UML switch“ byl mapován na fyzické síťové rozhraní „PC simulátoru“. Na zbývající 2 fyzická rozhraní „PC simulátoru“ byly mapovány uzly UML4 a UML5, podobně jako v případě konfigurace 1, resp. obr. 4.3.

Měření zpoždění konfigurace 2 zachycené v tab. 4.1 a grafu na obr. 4.5 proběhlo opět programem Ping. Měřena byla trasa mezi uzly UML1 a UML5.

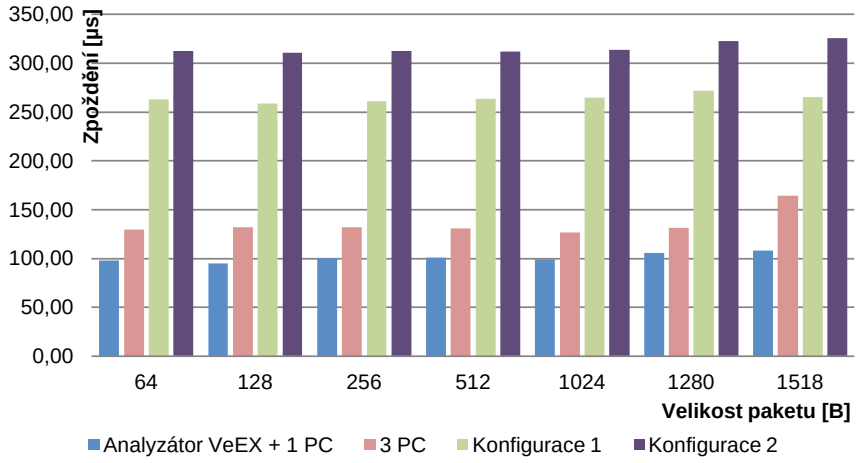


Obr. 4.4: VNUML konfigurace 2

Výsledky měření zaznamenané v tab. 4.1 jsou vyneseny do sloupcového grafu obr. 4.5. Z grafu je patrné, že zpoždění je pro různě velké pakety/rámce prakticky neměnné. Systém pracoval při měření stabilně. Současně je ale z grafu patrný zásadní vliv topologie, ale i metodiky měření na výsledky.

Původním předpokladem bylo, že měření provedená v rámci „konfigurace 1“, měření v konfiguraci „3 PC“ a měření analyzátorem VeEX (popis analyzátoru viz kap. 3.1.1) poskytnou přibližně stejné výsledky. Ve všech třech vyjmenovaných případech se jednalo o hvězdicovou topologii s centrálním prvkem „PC switch“. Konfigurace 2 posloužila jako potvrzení domněnky, že zásadní zpoždění do měření vnáší „PC simulátor“.

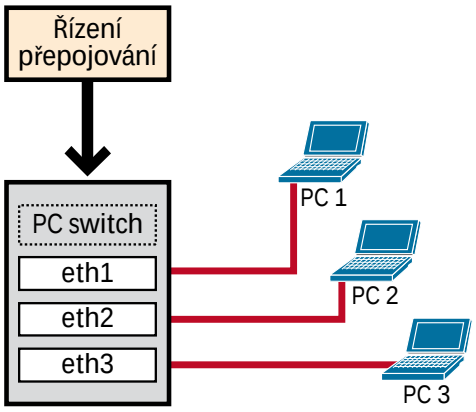
Při měření analyzátorem VeEX byla použita topologie odvozená od topologie na obr. 4.6. Centrálním prvkem byl „PC switch“, analyzátor VeEX byl jedním portem připojen k jeho rozhraní eth1, druhým k rozhraní eth2. Datový tok (náhodná data L2) vytěžující zátěží 30 % rozhraní byl generován rovněž analyzátorem VeEX a byl veden mezi rozhraním eth3 a rozhraními eth1 a eth2. Zajímavý je rozdíl zpoždění v případě měření analyzátorem VeEX a s použitím konfigurace „3 PC“, který je cca 30 μs s výjimkou velkých rámců 1 518 B, kde byl změřen rozdíl absolutního zpoždění $\Delta l_a = 56{,}2\ \mu s$.



Obr. 4.5: Vliv topologie VNUML modelu na zpoždění

Kvůli rozdílným výsledkům dosažených při použití „PC simulátoru“, bylo od jeho dalšího použití a vývoje upuštěno.

Měření předkládaná souhrnně v rámci kap. 3.1 byla provedena na výrazně jednodušší konfiguraci s pouze třemi samostatnými koncovými uzly (PC) tak, jak je zachyceno na obr. 4.6. Propojení bylo realizováno 1000Base-T, k měření výkonnosti byl použit program Ping a metodika popsaná v kap. 3.1. Emulátor „PC switch“ byl spuštěn na PC osazeném procesorem Intel CPU i7 920; 2,67 GHz; 6 GB RAM; HDD 10 000 RPM. Testování proběhlo na operačním systému Ubuntu 8.04. Závislost na výkonu CPU, propustnosti sběrnic apod. nebyla měřena, i když ji lze očekávat na základě měření „konfigurace 1“ a „konfigurace 2“. Provedené měření je pravděpodobně postiženo chybou v důsledku nemožnosti eliminovat v rámci systému „PC switch“ všechny procesy bez vztahu k provedeným měřením.



Obr. 4.6: Konfigurace emulátoru pro měření

Jako podpora popsaného emulátoru, resp. celého výzkumu, byl dále ve spolupráci s T. Pelkou vyvinut a sestaven síťový emulátor SimPP, jehož úkolem bylo zvoleným způsobem narušovat klíčové parametry QoS, viz kap. 1.1 a 1.2. Emulátor umožňuje přesně nastavit a do provozu vnést – zpoždění, ztrátovost a jitter. Emulátor posloužil při studiu vlivu vyjmenovaných parametrů na jednotlivé služby poskytované IP sítí. Emulátor, jeho vlastnosti a s ním dosažené výsledky jsou popsány v publikacích [109] a [110].

4.1.3 ZHODNOCENÍ

Obtížná opakovatelnost výsledků, závislost na vedlejších okolnostech – výkon PC, procesy operačního systému, propustnost sběrnic atd. ukázaly, že modely vytvořené v rámci systému VNUML lze použít spíše pro výukové potřeby. Vytvořené algoritmy sice lze přenést do praktické realizace, ale výkon navržených algoritmů prakticky nelze vzájemně porovnávat. Z popsaných důvodů byl výzkum na platformě VNUML zastaven.

4.2 TRÉNOVÁNÍ NEURONOVÉ SÍTĚ V REÁLNÉM PROVOZU

Proběhlo testování několika možností využít neuronovou síť k identifikaci toků a predikci jejich vývoje s využitím soběpodobnosti (samopodobnosti). Návazně pak měl být využít predikovaný tok k sestavení přepínací matice – rovněž s použitím neuronové sítě. Rozpoznávání jednotlivých toků a jejich zařazování do front na jejich základě se jevilo jako perspektivní. Postup by mohl umožnit identifikovat různé toky, které sdílí stejný protokol 7. vrstvy, např. HTTP.

Výzkum [111] prokázal, že síťový provoz má při dlouhodobém sledování kladný Lyapunův exponent – systém je chaotický, bez možnosti dlouhodobé predikce jeho chování.

Z krátkodobého pohledu je dle [111] chování síťového provozu predikovatelné. Výzkum směřoval k ověření skutečností popsaných v [111] na větším vzorku.

Ačkoli jsou různými institucemi dány k dispozici připravené vzorky provozu, např. [112], nebylo možné je využít – bylo třeba, aby kvůli přesnosti použité vzorky provozu a specifika implementace protokolů odpovídaly specifikům dostupných zařízení. V průběhu výzkumu bylo nashromážděno cca 1,21 TB zachyceného síťového provozu. Postupně se ale ukázalo, že takto uložená data jsou obtížně zpracovatelná, rozpoznatelná a importovatelná do MATLABu, zvoleného simulačního prostředí. Byl proveden pokus s importem cca 5MB pcap souboru, ale množina tréninkových dat byla příliš malá – neuronovou síť se nepodařilo natrénovat, *mse* nabýval hodnot na úrovni statistické chyby bez ohledu na to, kolika epochami trénování sítě prošlo. Postup byl jako neefektivní následně opuštěn. Zvolenými nástroji na dostupném hardwaru nebylo možné nashromážděných 1,21 TB zpracovat.

Potřebný počet konstant v neuronové síti by se pohyboval v řádech desítek milionů. Konvenční algoritmy řízení a rozpoznávání toků založených na vzorech, portech atd., viz kap. 1.2, se jeví jako efektivnější.

Shromažďování vzorků provozu (tréninkové množiny) proběhlo ve 2 fázích. Specifický provoz (vzorky VoIP, HTTP, SSH) byl zachycen pomocí TAPu LineEye LE-580FX a uložen do pcap souborů programem Wireshark. Široký nediferencovaný soubor vzorků obsahující jak selektivně zachycené protokoly, tak ostatní obecný provoz proběhl pomocí analyzátoru Endace EndaceProbe EP7010-PS-FC, který je výkonný, ale poskytuje jen omezené možnosti hloubkové paketové analýzy. Záměrem bylo získané soubory importovat v binární podobě do MATLABu… Pro další výzkum se ukázalo být efektivnější použití simulovaného provozu.

4.3 Modelovaný protokol

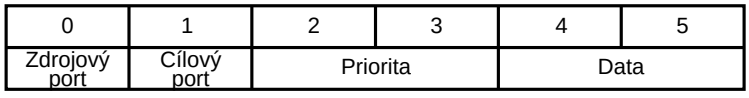
Pro účely modelování problematiky spojené s QoS byl navržen zjednodušený protokol, resp. paket. Při návrhu byla snaha, aby obsahoval všechna pole, která mohou mít na QoS vliv. Navržený paket byl inspirován protokolem IP, viz kap. 1.2.2 a 1.2.3, jeho cílem je tedy modelovat provoz na L3 RM ISO/OSI. Návrh obsahuje obdobu IP adresy zdroje a cíle, obdobu pole DSCP a datovou část.

Požadavkem na paket bylo, aby jeho hlavička měla konstantní velikost – to umožňuje zjednodušit model. Naopak část datová by měla mít velikost proměnlivou, což přiblíží charakteristikou a chováním paket vzorovému protokolu IP. Generátory paketů, viz kap. 5.2, by navíc měly umožnit konfigurovat datovou část tak, aby ve vybraných experimentech mohla být velikost paketu konstantní včetně datové části.

Pozn. Výhod paketu konstantní velikosti využívá např. ATM (Asynchronous Transfer Mode). Pro konstrukci přepínačů je mj. výhodné, že s použitím paketů, resp. rámců stejné velikosti, lze celý přepínač řídit synchronně. Synchronnost řízení je také jedním z důvodů zavedení buněk – particle, viz kap. 1.3.1.

4.3.1 Struktura protokolu

Schematický návrh nového paketu určeného pro použití v modelech je zachycen na obr. 4.7. Jeho hlavičku tvoří pole „Zdrojový port“, „Cílový port“ a „Priorita“.



Obr. 4.7: Model paketu

Zdrojový port je obdobou zdrojové IP adresy. Může nabývat celočíselných hodnot 1–4. Pole identifikuje port, resp. generátor paketů, ze kterého byl paket odeslán.

Cílový port je obdobou cílové IP adresy. Nabývá celočíselných hodnot 1–4, označuje port, na který je paket směrován.

Priorita – pole je složeno ze dvou částí, tak aby odpovídalo poli DSCP, viz kap. 1.2.2. První číslice dvouciferného čísla označující prioritu nabývá celočíselných hodnot 1–4 (reprezentuje prioritu), přičemž vyšší číslo znamená vyšší přednost. Druhá číslice je z celočíselného intervalu 1–3 a určuje pravděpodobnost zahození paketu. V případě pravděpodobnosti zahození znamená vyšší hodnota větší pravděpodobnost zahození. Hodnota celého pole „priorita" je počítána podle vztahu (4.1), kde P specifikuje prioritu (přednost) paketu a d označuje pravděpodobnost jeho zahození. Hodnoty pole odpovídají hodnotám DSCP AF, viz příloha A.4.

$$priorita = P \cdot 10 + d \tag{4.1}$$

Data – poslední pole modelového paketu je rovněž celočíselné. Pole zastupuje reálně přenášená data, přičemž pole má v binární reprezentaci proměnlivou velikost, ale může být definováno i staticky. Pole „Data" má kromě simulace reálného obsahu **význam identifikátoru paketu**. Ve většině simulací je pole naplňováno náhodnými čísly z intervalu 1–10 000. Pravděpodobnost nežádoucího zopakování paketu se stejnými poli „Zdrojový port", „Cílový port" a „Data" během jejich náhodného generování v požadovaném intervalu je malá. Jednotlivé pakety se shodným zdrojem a cílem lze díky poli „Data" snadno odlišit.

Navržený paket byl implementován do paketových generátorů vytvořených a implementovaných jako součást komplexního modelu přepínače.

5 KOMPLEXNÍ SIMULAČNÍ MODEL

Klíčovou částí výzkumu byla tvorba návrhu aktivního prvku (přepínače) a jeho modelu. Přepínač (switch) je systém, jehož fyzikální podstata byla zjednodušena – byl vytvořen zjednodušený komplexní model, pracující s diskrétním časem. Vytvořený model pracuje s pakety jako s kvanty. Simulační model byl vytvářen s cílem shrnout komplexně většinu proměnných, které mohou ovlivnit průchod paketu aktivním prvkem a umožnit jejich libovolnou modifikaci pro nalezení optimálních parametrů a řídicích algoritmů.

Porovnávání výkonnosti modelovaných algoritmů probíhalo buď jejich paralelním provozem v rámci jednoho modelu, nebo srovnáváním časových průběhů. V případě porovnávání časových průběhů musel model pracovat s předdefinovaným provozem tak, aby byly výstupy srovnatelné. Tato srovnávací metoda byla používána jen výjimečně – obvykle při identifikaci chyb v modelu. Výhodou softwarového modelu je možnost paralelně zařadit 2 a více celků a výstupy srovnat. Model takto umožňuje porovnávat de facto každou konstrukční část, čehož bylo při výzkumu často využíváno. Efektivitu algoritmů lze měřit také počtem simulačních kroků, který je potřeba k dosažení cíle. V konečném důsledku ale řešení vyžadující méně simulačních kroků nemusí znamenat, že řešení nebo algoritmus je efektivnější. O efektivitě rozhodně až finální implementace a také případná dostupnost ASIC.

5.1 POPIS MODELU

Modelován byl přepínač se 4 vstupními a 4 výstupními porty tak, jak bylo zmíněno v kap. 3.2. Přepínač byl navržen jako plně duplexní – tedy pro provoz s odděleným vysílacím a přijímacím kanálem. Provoz s poloviční duplexem není navrženým modelem podporován. Počet portů byl zvolen tak, aby byla současně zachována jednoduchost modelu a možnost simulovat všechny požadované jevy a situace a kolizní stavy.

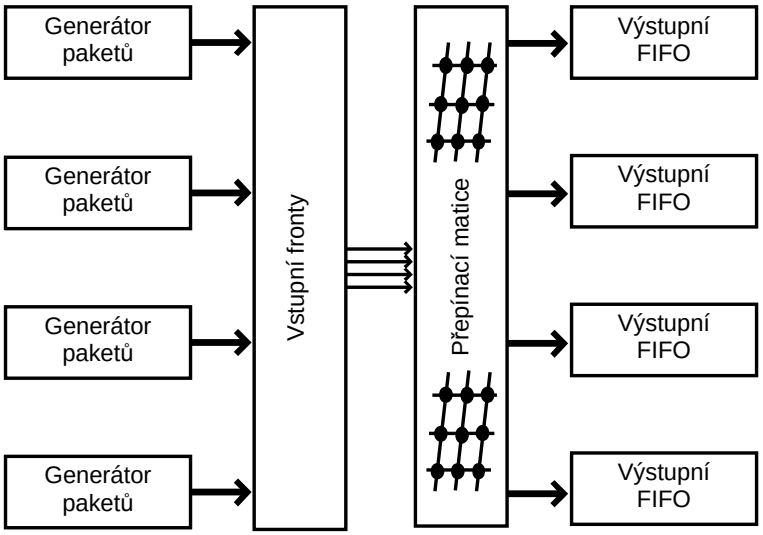
Navržený přepínač je částečně založen na konvenční struktuře popsané v rámci kap. 1.5 (obr. 1.12 a obr. 1.13). Model je navíc doplněn o generátory provozu, blok monitorování a nově navržené řízení spojovacího pole. Celý přepínač je navržen s ohledem na QoS a na minimalizaci negativních důsledků jeho činnosti na přenášený síťový provoz.

Unikátní je řízení spojovacího pole, které je řízeno neuronovou sítí „Feedforward Backpropagation“ a správa vstupních front, která v rámci komplexního modelu používá modifikovaný algoritmus MDRR, viz kap. 3.2.2.

Model byl vytvořen v MATLABu 2010b, především v Simulinku. Řídicí neuronová síť pak byla rovněž navržena v MATLABu s použitím „Neural Network Toolbox“.

Základní blokové schéma navrženého modelu je zachyceno na obr. 5.1. Pohled na schéma první vrstvy modelu, viz příloha E.1. Hlavními bloky přepínače jsou:

- generátory paketů,
- vstupní fronty,
- přepínací matice (včetně bloků řízení neuronovou sítí),
- a výstupní fronty (bloky výstupních front jsou na obr. 5.1 označeny jako „Výstupní FIFO“).

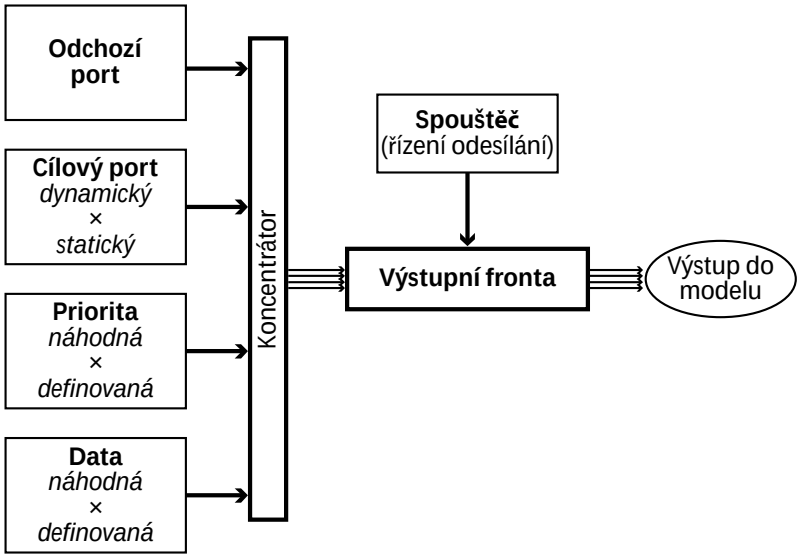


Obr. 5.1: Základní blokové schéma modelu

5.2 GENERÁTOR PAKETŮ

Model obsahuje 4 stejné paketové generátory. Každý z generátorů je zcela nezávisle konfigurovatelný. Základní blokové schéma generátoru je zachyceno na obr. 5.2. Generátory generují paket ve formátu popsaném v kap. 4.3.1.

Generátory jsou synchronizovány a inicializovány blokem „Spouštěč". Blok „Spouštěč" je uživatelsky konfigurovatelný a je také složen ze 4 subsystémů, přičemž může být každý ze 4 těchto subsystémů konfigurován samostatně a odlišně od ostatních. Tato vlastnost je výhodná pro simulaci sítí s různými charakteristikami propojenými modelovaným aktivním prvkem. Pakety mohou být generovány náhodně, nebo uživatelsky předdefinované. To, že v daném okamžiku není vyslán žádný paket, je modelováno vygenerováním paketu majícího všechna pole rovna 0. První vrstva modelovaného generátoru, viz příloha E.1.



Obr. 5.2: Schéma generátoru paketů

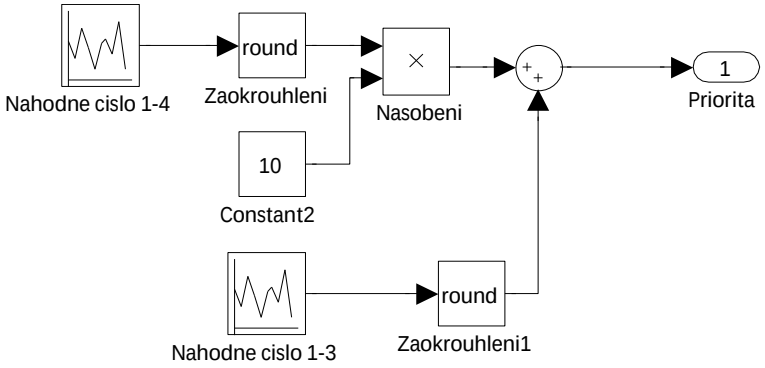
V bloku „Spouštěč" lze měnit jak rychlost generování paketu, tak také pravděpodobnost, se kterou budou generovány prázdné pakety, tedy kdy nebude na daném portu do systému vysílán paket. Přesné nastavení lze provést v bloku „perioda 0 paketu" – samostatně pro každý z paketových generátorů. Lze nastavit generování s uniformní pravděpodobností generování

prázdného paketu, nebo lze zvolit Gaussovo rozložení pravděpodobnosti. Schopnost externě řídit rychlost generování paketů pro každý z generátorů umožňuje v budoucnu modelovat i chování protokolů vyšších vrstev nebo pokročilých mechanismů zahazování paketů, jako např. Modified Tail Drop aj.

Přepínačem „náhodná perioda/linková rychlost“ lze vkládání prázdných paketů zastavit a testovat systém plnou linkovou rychlostí. Výjimečné výskyty prázdných paketů na výstupu generátoru v tomto režimu jsou zapříčiněny filtrací kolizních paketů – tedy těch, které mají shodný zdrojový a cílový port.

Na výstupu bloku „Generátor“ je ověřováno, zda se neshodují pole zdrojového a cílového portu. Vzhledem k tomu, že oba tyto porty mohou být (dle konfigurace) voleny náhodně, může ke shodě dojít. Protože pakety se shodným zdrojovým a cílovým portem jsou defektní a v reálném provozu se nevyskytují, jsou zahazovány (resp. nahrazovány nulovými pakety). Ačkoli lze v generátoru vytvářet pakety s libovolnou zdrojovou i cílovou adresou (tedy i mimo rozsah modelovaného přepínače), model ve své předkládané podobě tyto stavy neošetřuje.

Výpočet hodnoty pole „priorita“ je prováděn podle vztahu (4.1) tak, aby bylo co nejpodobnější poli DSCP AF, jak je pospáno v kap. 4.3.1. Každá ze dvou částí pole *P* a *d* může být definována uživatelsky, nebo může být volena náhodně. Náhodný generátor prioritní třídy je zachycen na obr. 5.3



Obr. 5.3: Blok náhodného generování prioritní třídy

Reálný ethernetový provoz může být z hlediska časování považován aktivním prvkem za pseudonáhodný. Pakety mohou být do aktivního prvku doručovány stochasticky – zpoždění paketů se může lišit. Kvůli simulaci tohoto jevu – jitteru – jsou pakety po vygenerování v rámci bloku „Generátor“ uloženy do fronty FIFO, dokud nejsou odeslány ven z generátoru v náhodném nebo uživatelsky definovaném čase.

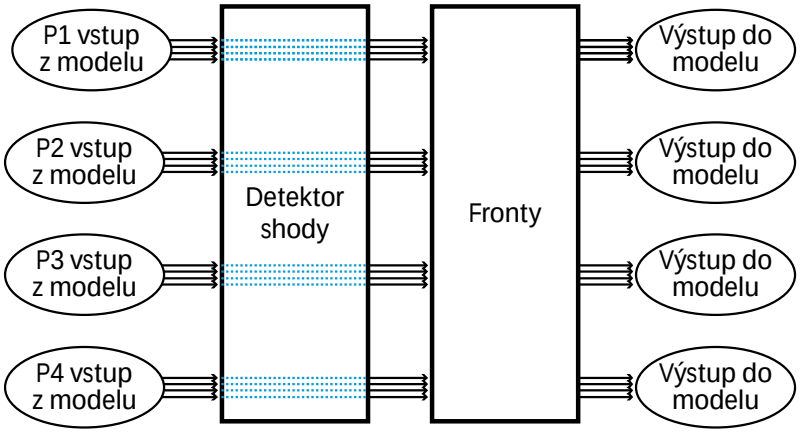
Podstatným problémem, který bylo třeba v rámci modelu vyřešit, byla detekce začátku a konce paketu, resp. rámce. Celá problematika je v rámci Ethernetu a TCP/IP vyřešena, ale z důvodů popsaných v kap. 4.3 nebylo praktické modelovat celou sadu protokolů. Vkládat do pro model vytvořeného protokolu preambuli se neosvědčilo při úvodních testech. Model musel na mnoha místech paketu provádět syntaktickou analýzu celého protokolu (parsing). Celý problém byl vyřešen s použitím paralelního (sběrnicového) systému datového přenosu v rámci modelu. V místech, kde je to možné, je celý provoz uložen do sběrnice (blokem Bus Creator), reprezentovaným v blokovém schématu (obr. 5.2) modulem „Koncentrátor“. V bloku, kde je třeba s některou z částí paketu pracovat, lze snadno tuto část vydělit (blok Bus Selector). Každé z polí protokolu je tedy v rámci celého modelovaného přepínače přenášeno důsledně odděleně. Ačkoli reálně (na binární vrstvě) mají pakety protokolu proměnlivou délku,

přenos sběrnicemi s nimi umožňuje pracovat jako s „kontextem“, který není v rámci takto řešeného přepínače třeba vytvářet.

5.3 VSTUPNÍ FRONTY

Blok „Vstupní fronty“ je klíčovým blokem řešícím problematiku priority provozu v rámci celého přepínače. Umožňuje volit na nejvyšší úrovni ze dvou možností správy vstupních front, resp. řešení kolizí paketů směřujících do stejného cíle. Zjednodušené blokové schéma základních součástí je zachyceno na obr. 5.4.

Součástí bloku vstupních front je podblok „Detektor shody“, jehož účelem je detekovat a označovat pakety se shodným cílem, tedy pakety kolidující. Podblok „Detektor shody“ pracuje s kopií původního paketu (na obr. 5.4 je volný průchod paketu blokem naznačen modrou čárkovanou čarou). Detektor shody označí kopie kolidujících paketů hodnotou –1 v poli „Priorita“. Pokud blok neidentifikuje kolizi paketů, jsou všechny pakety bezprostředně předávány do spojovacího pole (bloku „Přepínací matice“) a následně do výstupních front.

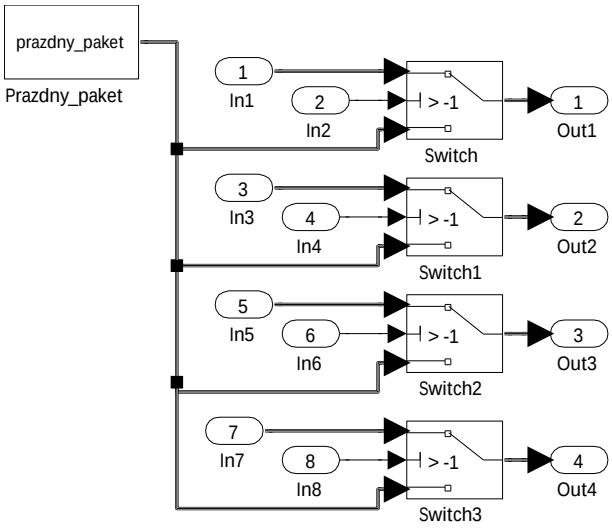


Obr. 5.4: Blokové schéma správy vstupních front

Pokud je detekována kolize, umožňuje komplexní model dva různé způsoby nakládání s kolidujícími pakety.

První možností, kterou lze v bloku řízení vstupních front volit pro případ kolize paketů, je zahození paketu s nastavenou nižší prioritou *P*. Pokud koliduje více paketů, je vždy propouštěn pouze paket s nejvyšší prioritou, ostatní jsou zahozeny (zahození je realizováno stejně, jako v případě zahazování v generátoru, tedy nahrazením paketu prázdným paketem). Schéma je zachyceno na obr. 5.5. Jedná se o rychlé a transparentní řešení, ale se zbytečnou ztrátovostí paketů. Tato metoda řešení kolizí je implementována v bloku „Zahazovani shody“.

Druhá možnost zacházení s kolidujícími pakety v bloku vstupních front je, ve srovnání s volbou první, sofistikovanější. Cílem návazných podbloků je plně podporovat QoS. Dva a více pakety s detekovanou kolizí jsou odeslány do podbloku „Řazení QoS“, který je jedním z podbloků bloku „Fronty“ na obr. 5.4.



Obr. 5.5: Zahazování kolizních paketů

V rámci podbloku „Řazení QoS“ jsou podle pravidel popsaných v kap. 3.2.2 ukládány pakety do front. Bloky „QoS razeni1–4“ rozdělí pakety do skupin. Snímek bloku „QoS razeni1“, viz příloha E.3.

Pro každý vstupní port je definována jedna skupina front pro uložení 8 paketů. Pokud je ve frontě více než 8 paketů, všechny nově příchozí pakety jsou zahozeny (mechanismus Tail Drop, viz kap. 1.3.4 a 3.2.2). Každá skupina front je složena z jednotlivých front – pro každou prioritní třídu je dedikována 1 samostatná fronta. Do front jsou pakety rozdělovány podle priorit.

Pozn. Protože na úrovni simulinkového modelu není možné do FIFO fronty uložit obsah sběrnice (a přečíst jej zpět jako obsah sběrnice), jsou pro jednotlivé části paketu vyčleněny samostatné fronty. Každý z bloků „FIFO dle priorit1–4“ na schématu v příloze E.3 zastupuje jedno pole paketu. Např. do bloku „FIFO dle priorit1“ je ukládán odchozí port. V rámci bloků „FIFO dle priorit“ je definován subsystém se skupinou front rozdělených dle prioritních tříd – viz schéma v příloze E.4. Celek je zapisován a čten centrálně a synchronně. K odesílání paketů mimo pořadí nebo k problémům s odtržením hlavičky od datové části nemůže dojít.

V případě, že je výstupním rozhraní (port) volné a připravené pro odeslání paketu, jsou čteny a postupně odesílány pakety z front. Základním algoritmem pro čtení paketů z front je algoritmus se striktní prioritou SMDRR, viz kap. 3.2.2 a obr. 3.6 – začíná s odesíláním paketů z fronty s nejvyšší prioritou. V případě, že je tato fronta prázdná, algoritmus pokračuje s frontou s nižší prioritou atd. Blokem následujícím za vstupními frontami je spojovací pole.

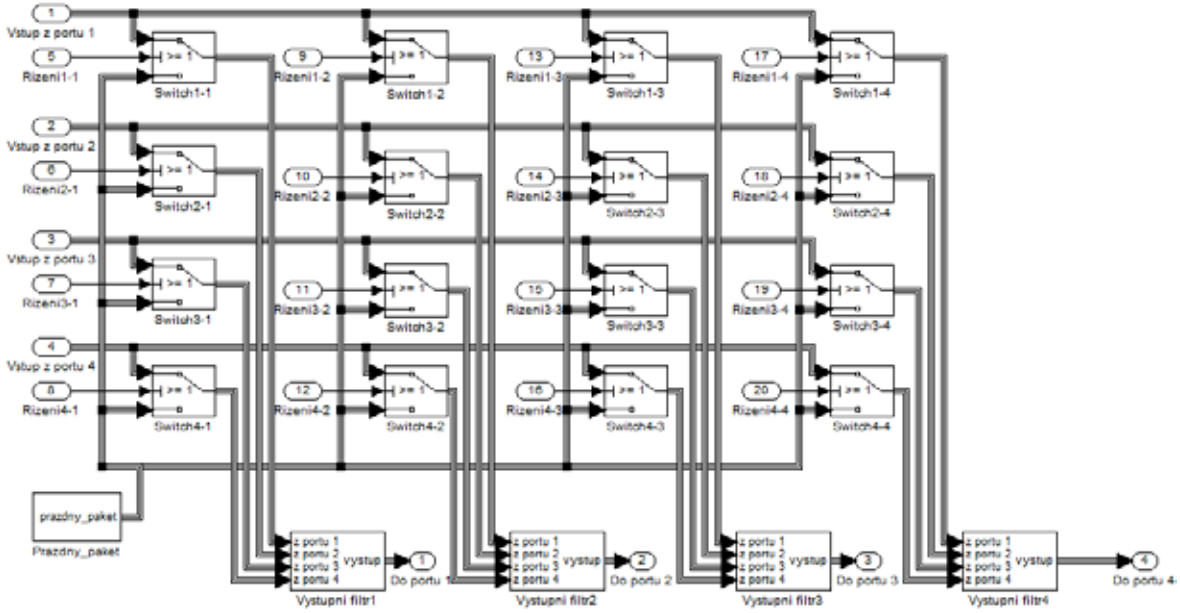
5.4 SPOJOVACÍ POLE S ŘÍZENÍM

Blok nazvaný v modelu a v příloze E.1 „Krizovy_spinac_s_rizenim“ se skládá ze dvou hlavních částí – řízení (generátoru konfigurační matice) a spojovacího pole – křížového spínače. Řešení zapojení křížového spínače v modelu je zachyceno na obr. 5.6. Spojovací pole je modelováno jako pole 4×4 jednoduchých externě řízených spínačů.

5.4.1 POPIS SPOJOVACÍHO POLE

Jak je patrné z obr. 5.6, na vstupy jednotlivých spínačů jsou přiváděny sběrnice (pakety) – vždy stejný vstupní paket na jeden řádek spínačů. Každý ze sloupců spojovacího pole směřuje

do jednoho výstupního portu. Blok „Vystupni filtr1–4“ slouží k identifikaci platného paketu (odlišení od prázdných paketů) a sloučení 4 výstupů do jednoho cílového směřujícího do výstupní fronty.



Obr. 5.6: Model křížového spínače

Součástí bloku „Krizovy_spinac_s_rizenim“ je subsystém monitoringu stavu spojovacího pole. Subsystém monitoringu lze účinně využít pouze, pokud je model, resp. simulace, provozován v pseudoreálném čase, viz kap. 5.7. V tom případě lze sledovat plynule podobu konfigurační matice a stavy všech vstupů a výstupů spojovacího pole.

5.4.2 *Řízení spojovacího pole*

Prostřednictvím vstupů „Rizeni“ každého ze spínačů na obr. 5.6 je celé spojovací pole ovládáno tzv. konfigurační maticí **C**. Konfigurační matice je generována v každém simulačním kroku. Může být generována jak konvenčním algoritmem, tak umělou neuronovu sítí.

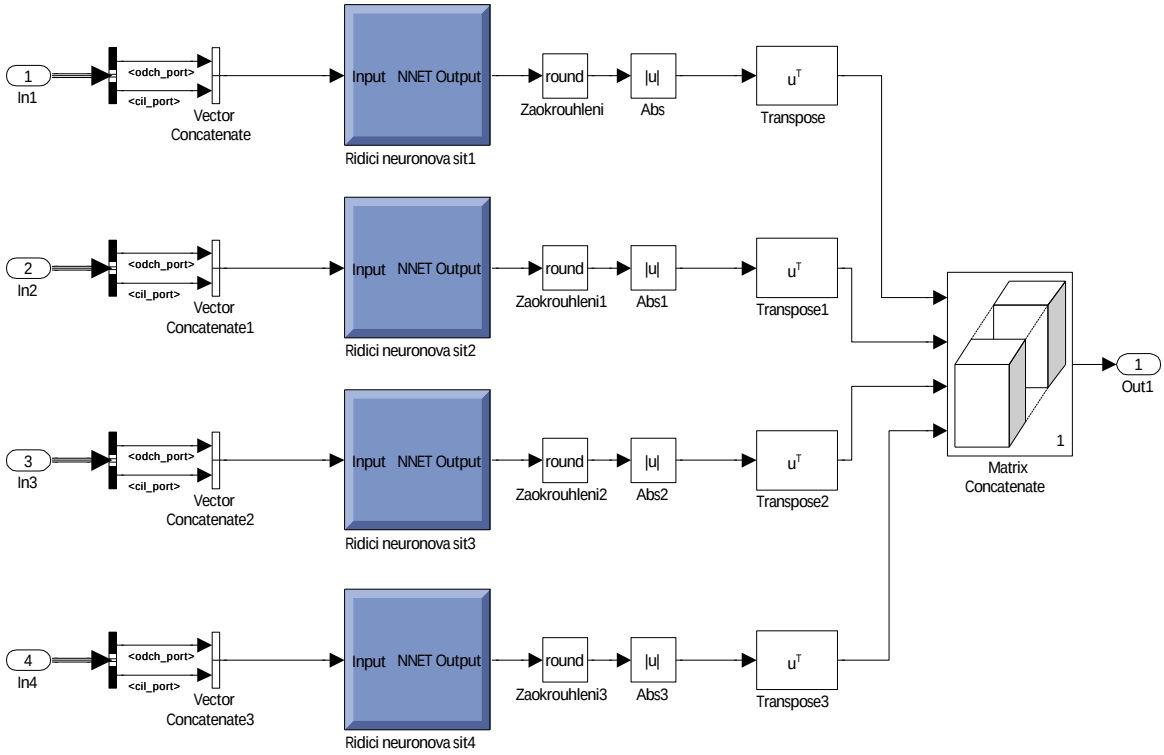
Konvenční přepínače jsou obvykle řízeny explicitně programováním definovaným algoritmem, který řídí síťový procesor a další součásti aktivního prvku. Navržený aktivní prvek je řízen alternativně – neuronovou sítí, viz kap. 3.3. Matice (5.1) je příkladem generované konfigurační matice.

Pozn. Součástí řídicí struktury, resp. bloku „Krizovy_spinac_s_rizenim“, je také subsystém nazvaný „Parser_konf_mat“. Jeho úlohou je přečíst z konfigurační matice **C** jednotlivé její členy odpovídající spínačům spojovacího pole a následně je odeslat na příslušný řídicí vstup daného spínače.

$$\mathbf{C} = \begin{pmatrix} 0 & 1 & 0 & 0 \\ 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 \\ 0 & 0 & 1 & 0 \end{pmatrix} \tag{5.1}$$

Přepínač je navržen tak, aby pracoval zcela paralelně, z čehož by v ideálním případě plynulo, že s rostoucím počtem portů nebude klesat výkon. V praxi je počet portů promítnut do velikosti paralelních řídicích neuronových sítí, z čehož plyne pokles výkonu s přibývajícím počtem portů. Další pokles výkonu je závislý na zastoupení kolidujících paketů v provozu.

Paralelní struktura návrhu je zachycena na obr. 5.7, který zobrazuje blok „Gen_konf_mat“. Čtyři řídicí neuronové sítě jsou na obr. 5.7 vyznačeny modře. Pro každou vstupní frontu je vyčleněna jedna samostatná neuronová síť.



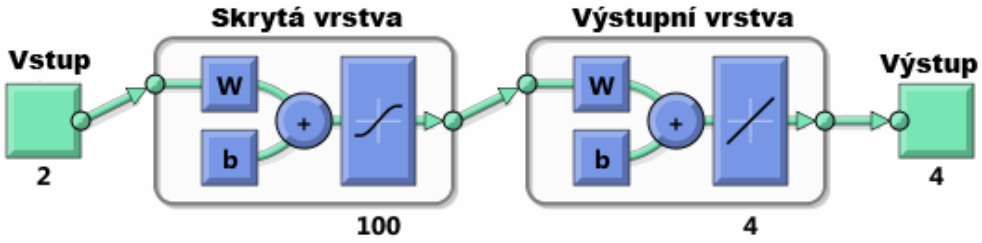
Obr. 5.7: Struktura bloku řídicích neuronových sítí

Vektor vstupující do neuronové sítě **P** se skládá ze dvou částí (čísel) – zdrojového a cílového portu. Výstupní (cílový) vektor **T** je složen ze 4 binárních čísel. Příkladem vstupního a výstupního vektoru jsou matice (5.2).

$$\mathbf{P} = \begin{pmatrix} 4 \\ 3 \end{pmatrix}; \mathbf{T} = \begin{pmatrix} 0 \\ 0 \\ 1 \\ 0 \end{pmatrix} \tag{5.2}$$

Struktura neuronové sítě je odvozena od otestované struktury popsané v rámci kap. 3.3.3. Síť je typu „Feedforward Backpropagation“, viz kap. 3.3.2. Volba, resp. vhodnost použití tohoto typu sítě byla potvrzena výsledky testů popsanými v kap. 3.3.3 a rovněž během trénování sítě pro řešení přepínání paketů. Ve skryté vrstvě sítě bylo použito 100 neuronů, ze 4 neuronů se pak skládá výstupní vrstva, viz obr. 5.8.

Pozn. Řídicí neuronová síť byla po natrénování nástrojem „gensim“ konvertována do Simulinku. Matlab skripty pro něj byly mj. také použity pro vývoj skriptů k testování funkčnosti celého modelu.



Obr. 5.8: Vizualizace modelu řídicí neuronové sítě

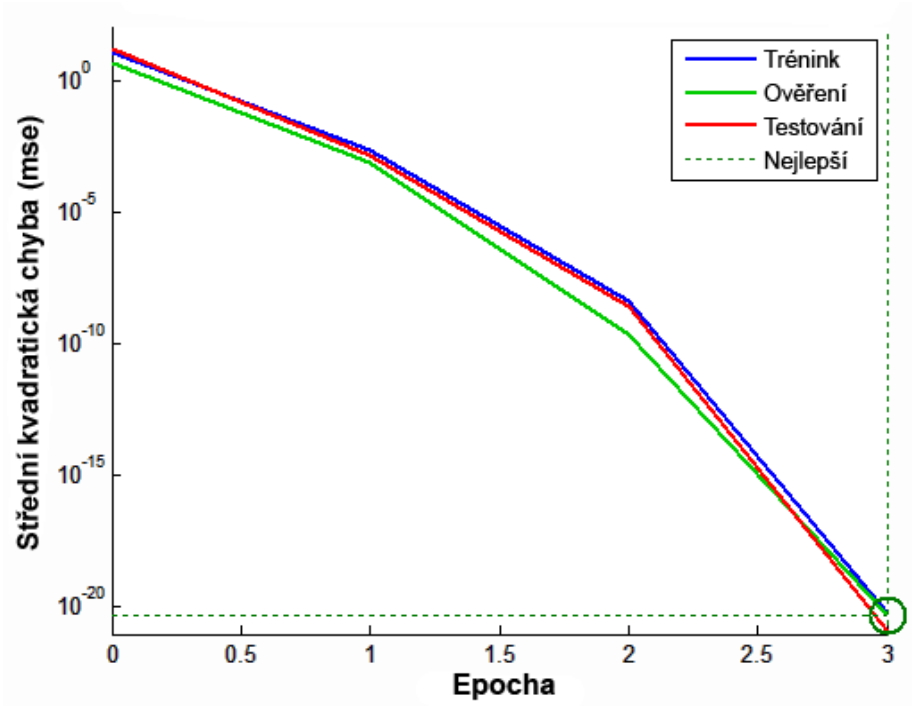
Ukázková část tréninkové množiny následuje.

```
p = [4 3 2 1, 4 3 2 1, …,4 4 4 3…;
     3 2 1 4, 2 1 4 3, …,1 2 3 1…];

t = [0 0 1 0, 0 1 0 0, …,1 0 0 1…;
     0 1 0 0, 1 0 0 0, …,0 1 0 0…;
     1 0 0 0, 0 0 0 1, …,0 0 1 0…;
     0 0 0 1, 0 0 1 0, …,0 0 0 0…];
```

Tréninková množina je rozdělena:

- 90 % celé tréninkové množiny je použito k trénování,
- 5 % tréninkové množiny je použito k ověřování (validaci),
- 5 % tréninkové množiny použito k testování.

Vytvořená neuronová síť rychle konvergovala k požadované střední kvadratické chybě mse (mean squared normalized error, viz kap. 3.3.3, rovnice (3.10)). Po třech epochách byla dosažena požadovaná $mse = 4{,}306 \cdot 10^{-21}$. Vývoj mse v průběhu trénování neuronové sítě je na obr. 5.9.



Obr. 5.9: Průběh trénování sítě – vývoj mse

Další bloky, které jsou součástí bloku „Gen_konf_mat“, slouží k zaokrouhlení výstupu neuronové sítě na celá čísla a vytváří z jednotlivých konfiguračních vektorů konfigurační matici pro řízení spojovacího pole jako celku.

5.5 Výstupní FIFO fronty

Každému výstupnímu portu je přiřazena jedna výstupní FIFO fronta – v rámci obr. 5.1 označena „Výstupní FIFO“. Výstupní FIFO fronty svým určením odpovídají HW frontám popsaným v kap. 1.3.1. Velikost fronty je nastavena na 8 paketů, stejně jako v případě vstupních FIFO front.

Pakety jsou ukládány do výstupních FIFO a čteny pouze tehdy, je-li cílový port na opozičním zařízení připraven odesílaný paket přijmout. To je modelováno samostatným řídicím vstupem na každé frontě, na který lze připojit samostatný spouštěč. Protože většinou byly řešeny úlohy spojené s QoS a řízením spojovacího pole, bylo nežádoucí, aby na výstupních FIFO frontách docházelo ke ztrátám paketů nebo dalšímu zpožďování provozu. Proto byly obvykle provozovány výstupní fronty v synchronním režimu – do bloku monitorování byl odeslán stejný počet paketů, jako byl přijat na vstupu.

5.6 Výsledky a testování

Navržený model přepínače je vybaven testovacími a vizualizačními nástroji sloužícími pro ověření výsledků simulací. Kromě bloku monitorování, který je součástí bloku „Krizovy _spinac_s_rizenim“, viz kap. 5.4.1, je v modelu i další monitorovací blok umístěný přímo v první vrstvě modelu, viz obr. – příloha E.1, označený „Monitoring“.

Podrobným ověřovacím nástrojem v bloku „Monitoring“ je „srovnávací tabulka“. Srovnávací tabulka (obr. 5.10) je rozdělena na 2 barevně oddělené části – zelená část jsou pakety odeslané z paketových generátorů do přepínače (G1–4). Červená část srovnávací tabulky identifikuje pakety po přepnutí (odeslání) do výstupní FIFO (C1–4), resp. na výstup přepínače.

Pozn. V rámci konstrukce modelu moduly „Gen1–4“ a „Cil1–4“ odesílají obsah všech generovaných a výstupních paketů do pracovního prostoru Matlabu. Vyhodnocení a konstrukce srovnávací tabulky je zajištěno skriptem „`vyhodnoceni_v3.m`“. Finální grafická podoba je tabulce následně dána po importu do Excelu.

Dvě červeně vyznačené oblasti na obr. 5.10 jsou pohledem na jeden paket se strukturou, která je popsána v kap. 4.3.1 a která je znázorněna na obr. 4.7. Paket je zachycen v okamžiku výstupu z generátoru a znovu na výstupu z přepínače (po průchodu spojovacím polem). Řádky buněk odpovídají polím paketu schematicky znázorněnému na obr. 4.7:

- první řádek – zdrojový port,
- druhý řádek – cílový port,
- třetí řádek – priorita,
- čtvrtý řádek – data.

Horizontální osa na obr. 5.10 je osa časová a představuje ukázku 12 po sobě následujících kroků. Paket znázorněný červeně na obr. 5.10 je odesílán z generátoru ve druhém kroku. Stejný paket je ve třetím kroku simulace odesílán z přepínače.

G1	0	0	0	0	1	1	0	0	1	1	0	0
	0	0	0	0	2	2	0	0	3	4	0	0
	0	0	0	0	32	42	0	0	22	11	0	0
	0	0	0	0	1311	8716	0	0	931	9036	0	0
G2	0	2	0	2	0	0	0	2	2	0	0	0
	0	3	0	3	0	0	0	1	3	0	0	0
	0	23	0	11	0	0	0	23	42	0	0	0
	0	3701	0	1506	0	0	0	9126	4431	0	0	0
G3	0	3	0	0	0	0	3	0	3	3	0	0
	0	4	0	0	0	0	1	0	4	2	0	0
	0	41	0	0	0	0	22	0	23	12	0	0
	0	1401	0	0	0	0	3421	0	7831	4036	0	0
G4	0	4	0	0	0	0	4	4	0	4	0	4
	0	3	0	0	0	0	3	2	0	2	0	3
	0	33	0	0	0	0	33	33	0	43	0	42
	0	5201	0	0	0	0	4721	3726	0	2936	0	6746
C1	0	0	0	0	0	0	3	2	0	0	0	0
	0	0	0	0	0	0	1	1	0	0	0	0
	0	0	0	0	0	0	22	23	0	0	0	0
	0	0	0	0	0	0	3421	9126	0	0	0	0
C2	0	0	0	0	1	1	0	4	0	4	3	0
	0	0	0	0	2	2	0	2	0	2	2	0
	0	0	0	0	32	42	0	33	0	43	12	0
	0	0	0	0	1311	8716	0	3726	0	2936	4036	0
C3	0	4	2	2	2	0	4	0	2	2	1	4
	0	3	3	3	3	0	3	0	3	3	3	3
	0	33	23	11	23	0	33	0	42	23	22	42
	0	5201	3701	1506	3701	0	4721	0	4431	3701	931	6746
C4	0	3	0	0	0	0	0	0	3	1	0	0
	0	4	0	0	0	0	0	0	4	4	0	0
	0	41	0	0	0	0	0	0	23	11	0	0
	0	1401	0	0	0	0	0	0	7831	9036	0	0

Obr. 5.10: Srovnávací tabulka

Nulové pakety v některých buňkách indikují prázdný paket, resp. absenci paketu v daném simulačním kroku.

Některé odeslané pakety jsou detekovány po více než jednom simulačním kroku. Příčinou je aktivní systém zajišťující QoS a bránící kolizím paketů. Příklad tohoto druhu paketů je na obr. 5.10 vyznačen modrou barvou. Dva pakety se stejným cílem jsou odesílány z generátorů 1 a 2. Paket odeslaný z generátoru 2 má prioritu 42, tedy vyšší než paket pocházející z generátoru 1 a nesoucí prioritu 22. To je příčinou toho, že paket z generátoru 2 je doručen před paketem pocházejícím z generátoru 1.

Kromě podrobného zobrazení formou tabulky jsou do simulinkového modelu zabudovány statistické výstupy odesílané ke zpracování skriptem „`vyhodnoceni_v3.m`“ do MATLABu. Statistiky jsou sbírány na výstupech generátorů a na cílových portech. Statistiky jsou sbírány v rámci bloku „Blok pocitadel paketu“, který je tvořen 32 subsystémy. Úkolem je filtrovat a sčítač počty vygenerovaných paketů všech priorit. Obdobně jsou zpracovávány pakety přijaté, resp. pakety zaslané do výstupních rozhraní po průchodu přepínačem.

Měření jsou zcela nezávislá jak na zmíněné vygenerované „srovnávací tabulce“, tak na zbytku modelu. Počty vygenerovaných a ztracených paketů lze tedy kontrolovat na několika místech a různými nezávislými způsoby. Výsledky získaných měření jsou skriptem „`vyhodnoceni_v3.m`“ zpracovány (skript je třeba spustit samostatně). Příklad výstupu vyhodnocovacího skriptu:

```
Port C1: ztraceno bylo 1 z 10 odeslanych. Ztratovost 10 %
Port C1: nedoruceno P1 0; P2 0; P3 1; P4 0
```

```
-----
Port C2: ztraceno bylo 4 z 23 odeslanych. Ztratovost 17.3913 %
Port C2: nedoruceno P1 0; P2 2; P3 2; P4 0
-----
Port C3: ztraceno bylo 1 z 16 odeslanych. Ztratovost 6.25 %
Port C3: nedoruceno P1 0; P2 1; P3 0; P4 0
-----
Port C4: ztraceno bylo 3 z 14 odeslanych. Ztratovost 21.4286 %
Port C4: nedoruceno P1 0; P2 2; P3 1; P4 0
```

V uvedeném případě byl modelován průchod paketů přepínačem uvedených v tab. 5.3, kde:

- Z – označuje zdrojový (odchozí port).
- C – označuje cílový port.
- P – označuje prioritu.
- D – označuje data.

Výsledky jsou zpracovány v tab. 5.1 Z uvedeného příkladu je zřejmé, že ve všech případech došlo na cílových portech ke ztrátě paketů. Zajímavé je ale rozložení – které pakety byly zahozeny. V předloženém příkladě byl v modelu použit algoritmus SMDRR navržený a popsaný v kap. 3.2.2. Ztrátovost paketů s nejvyšší prioritou byla na všech portech nulová. Fronty s nižší prioritou (P2) ztratily 1–2 pakety.

Vzhledem k charakteru navrženého algoritmu je ztrátovost na portech s prioritou 3 a 4 očekávaná. Vysoká celková ztrátovost na všech portech je dána vysokým zatížením vstupních front – aplikuje se „Tail Drop“.

Cílový port č.	Odesláno pak.	Ztraceno celkem	Ztrátovost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	10	1	10,0	0	0	1	0
2	23	4	17,4	0	2	2	0
3	16	1	6,3	0	1	0	0
4	14	3	21,4	0	2	1	0

Tab. 5.1: Měření průchodu paketů přepínačem SMDRR

Pro srovnání byla sada paketů, viz tab. 5.3, zpracována druhým z bloků vstupních front, tedy „Zahazovani shody“. Výsledky po zpracování jsou uvedeny v tab. 5.2.

Oproti měření uvedenému v tab. 5.1 je patrná ztrátovost i u paketů s prioritou P1 – byly ztraceny 2 pakety směřující do portu 2 a 2 pakety směřující do portu 3. Obecně lze – v souladu s očekáváním – pozorovat rovnoměrnější rozložení ztrátovosti mezi jednotlivé prioritní třídy.

Cílový port č.	Odesláno pak.	Ztraceno celkem	Ztrátovost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	10	1	10,0	0	1	0	0
2	23	2	8,7	2	0	0	0
3	16	7	43,8	2	3	2	0
4	14	0	0,0	0	0	0	0

Tab. 5.2: Měření průchodu paketů přepínačem bez QoS

Pořadí	Generátor 1				Generátor 2				Generátor 3				Generátor 4			
	Z	C	P	D	Z	C	P	D	Z	C	P	D	Z	C	P	D
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	1	2	22	5401	0	0	0	0	3	4	12	7601	4	2	32	1201
3	0	0	0	0	2	3	22	6702	0	0	0	0	4	3	21	802
4	0	0	0	0	0	0	0	0	0	0	0	0	4	2	42	5606
5	0	0	0	0	0	0	0	0	3	2	23	2211	0	0	0	0
6	0	0	0	0	0	0	0	0	3	4	42	9016	0	0	0	0
7	1	4	32	4821	2	4	22	1321	3	1	31	8921	4	2	41	2521
8	0	0	0	0	0	0	0	0	3	1	31	3726	0	0	0	0
9	0	0	0	0	2	4	23	4131	3	1	31	831	0	0	0	0
10	0	0	0	0	0	0	0	0	3	2	33	8736	0	0	0	0
11	1	3	12	4841	2	4	11	8241	0	0	0	0	4	2	21	5441
12	1	4	33	1446	0	0	0	0	3	2	31	1646	0	0	0	0
13	1	2	33	3851	0	0	0	0	0	0	0	0	4	2	31	7951
14	1	3	43	7356	0	0	0	0	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
17	0	0	0	0	2	3	21	3471	3	2	12	3971	0	0	0	0
18	0	0	0	0	0	0	0	0	0	0	0	0	4	3	13	6976
19	1	3	23	1881	0	0	0	0	3	4	41	1981	0	0	0	0
20	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
21	1	3	13	2891	0	0	0	0	3	2	31	1591	4	1	33	191
22	0	0	0	0	2	1	32	5396	0	0	0	0	0	0	0	0
23	0	0	0	0	0	0	0	0	0	0	0	0	4	2	31	1401
24	0	0	0	0	0	0	0	0	3	1	21	4306	0	0	0	0
25	1	4	32	7511	2	3	32	4311	3	2	32	6911	0	0	0	0
26	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
27	1	2	41	6021	0	0	0	0	3	2	31	8421	4	3	22	9321
28	1	2	22	5326	0	0	0	0	0	0	0	0	0	0	0	0
29	1	2	32	2531	0	0	0	0	3	2	31	3031	0	0	0	0
30	0	0	0	0	0	0	0	0	3	2	22	2336	4	2	31	8836
31	1	4	42	6041	2	4	31	4941	3	4	21	7841	4	3	13	5141
32	0	0	0	0	2	1	42	3746	0	0	0	0	4	1	32	5446
33	1	4	22	7451	0	0	0	0	0	0	0	0	0	0	0	0
34	0	0	0	0	2	4	22	7456	0	0	0	0	4	1	42	9656
35	0	0	0	0	0	0	0	0	0	0	0	0	4	3	22	6461
36	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
37	0	0	0	0	0	0	0	0	3	2	32	7571	4	1	23	6871
38	0	0	0	0	0	0	0	0	0	0	0	0	4	3	33	2676
39	1	3	32	2781	0	0	0	0	0	0	0	0	4	3	12	7281
40	0	0	0	0	2	3	11	6086	0	0	0	0	4	2	23	2386

Tab. 5.3: Příklad provozu zpracovaného přepínačem

Při měřeních, viz tab. 5.1 a tab. 5.2 bylo vygenerováno vždy celkem 63 paketů (pakety jsou generovány a přijímány porty paralelně) ve 20 taktech, resp. jednotkách simulačního času s velikostí kroku 0,1, viz kap. 5.7. Během těchto 20 taktů model umožňuje vygenerovat a přijmout maximálně 40 paketů (reálně je simulací, resp. generátory, vygenerováno 41 skupin paketů – poslední 41. skupina ale není zpracována).

Simulováno bylo zatížení přepínače náhodným provozem – bylo celkem odesláno 1 464 paketů při 1 000 simulačních taktech. Výsledek je zachycen v tab. 5.4. Z výsledků měře-

ní uvedených v tab. 5.4 je opět patrná práce algoritmu SMDRR – nejméně jsou zahazovány pakety s nejvyšší prioritou. Protože ale byly při tomto měření odchozí pakety odesílány náhodně, nelze vyloučit zahazování mechanismem „Tail Drop“. Proto jsou počty zahozených paketů v polovině případů větší v prioritní třídě 3 oproti prioritě 4.

Cílový port č.	Odesláno pak.	Ztraceno celkem	Ztrátovost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	526	293	55,7	14	20	150	109
2	227	131	57,7	19	32	17	63
3	211	157	74,4	12	57	72	16
4	500	260	52,0	8	10	105	137

Tab. 5.4: Měření průchodu paketů přepínačem SMDRR – 1 000 simulačních taktech

Pokud je snížena rychlost generování paketů (úpravou periody v bloku „Spoustec_g“), úměrně dochází ke snížení ztrátovosti. Pokud je perioda snížena na polovinu – oproti příkladu uvedeném v tab. 5.1, pokles ztrátovosti je patrný z tab. 5.5.

Ztráta 2 paketů priority P2 směřujících na port C3 byla identifikována ve „srovnávací tabulce“ jako důsledek 5 paketů směřujících na port C3 – 2 + 2 + 1 ve 3 simulačních taktech.

Cílový port č.	Odesláno pak.	Ztraceno celkem	Ztrátovost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	2	0	0,0	0	0	0	0
2	14	0	0,0	0	0	0	0
3	14	2	14,3	0	2	0	0
4	20	2	10,0	0	0	0	2

Tab. 5.5: Měření průchodu paketů přepínačem SMDRR – 50% zatížení

Ovlivnit ztrátovost navrženého přepínače mohou rovněž délky vstupních softwarových front. V kap. 1.3 byl vliv délky front na ztrátovost zmíněn. Prodloužením softwarových front lze snížit ztrátovost, ale zásadním způsobem může dojít k nárůstu zpoždění. Zpoždění lze v rámci modelovaného přepínače pozorovat nejlépe v rámci „srovnávací tabulky“.

Prodloužení vstupních softwarových front bylo modelováno – výsledek je uveden v tab. 5.6. Jako srovnávací provoz byl opět použit tok zachycený v tab. 5.3. Prioritním třídám P1 a P2 byly délky front ponechány, jak bylo zmíněno v kap. 5.3 délky 8. Fronty P3 a P4 byly prodlouženy na 10 – to odpovídá ¼ všech možných vygenerovaných paketů (při délce simulace 20 simulačních taktů, která byla použita).

Cílový port č.	Odesláno pak.	Ztraceno celkem	Ztrátovost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	10	1	10,0	0	0	1	0
2	23	4	17,4	0	2	2	0
3	16	1	6,3	0	1	0	0
4	14	0	0	0	0	0	0

Tab. 5.6: Měření průchodu paketů přepínačem SMDRR – prodloužení front

Ztrátovost je závislá také na rozložení provozu – pokud je provoz generován náhodně, statisticky významná jsou až měření provedená na delších simulacích. V případě uvedeném v tab. 5.6 se na výsledku negativně projevil důsledek krátké simulace (cílový port č. 3) – pakety zůstávají uloženy ve frontách – nejsou ztraceny, ale protože simulace je po 20 simulačních taktech ukončena a pakety nebyly doručeny, jsou mezi ztracené, resp. nedoručené počítány.

Předejít problému lze buď prodloužením simulace, kdy bude délka simulace výrazně delší než je délka front, podobně, jako je tomu u reálného provozu a tím se ztrátovost posledních paketů stane statisticky nevýznamná nebo lze manuálně definovat odesílání pouze např. do poloviny délky simulace (a následně odesílat pouze prázdné pakety), tak aby se fronty vyprázdnily v průběhu simulace. Pro manuální analýzu chyb „srovnávací tabulkou" je tato metoda vhodnější.

Cílový port č.	Odeslá-no pak.	Ztraceno celkem	Ztráto-vost [%]	Ztraceno P 1	Ztraceno P 2	Ztraceno P 3	Ztraceno P4
1	3	0	0,0	0	0	0	0
2	11	0	0,0	0	0	0	0
3	7	0	0,0	0	0	0	0
4	8	0	0,0	0	0	0	0

Tab. 5.7: Měření průchodu paketů přepínačem SMDRR – zkrácení odesílání paketů

Výsledky měření potvrdily funkčnost navrženého přepínače. Při srovnání navrženého řešení s výsledky získanými měřením zpracovaným v kap. 3.1.4 je třeba zohlednit jak metodiku měření, tak celkovou architekturu aktivního prvku. Navržený a simulovaný aktivní prvek dokázal pracovat bez ztrát. Při nastavování a provozu simulace je ale třeba zohlednit zmíněná specifika vytvořeného modelu.

5.7 NASTAVENÍ SIMULAČNÍHO ALGORITMU

Na první vrstvě RM OSI lze na přenos Ethernetu pohlížet jako na přenos signálu proměnného, ale spojitého v čase. Aby byl simulační model maximálně transparentní, na nejnižších vrstvách pracuje simulační model rovněž se spojitým časem.

Důležitou částí modelu, kterou bylo třeba promyšleně nastavit, je nastavení konfiguračních parametrů simulace. Špatně zvolené parametry mohou vést k nepravdivým výsledkům nebo chybám při simulaci navrženého modelu.

Simulink, v použité verzi, umožňuje používat integrační metody s konstantním pevně daným krokem (fixed-step) nebo zvolit proměnnou velikost kroku (variable-step). V rámci obou lze volit konkrétní metodu řešení (resp. integrační metodu).

Řešení s konstantním vzorkovacím krokem nabízí metody řešení [113]:

- diskrétní (discrete) – pro systémy s diskrétním časem,
- dvě varianty Darmont–Prince integrační metody (ode8 a ode5),
- metodu Runge–Kutta (ode4),
- Bogacki–Shampine (ode3),
- Heun (ode2),
- Eulerovu metodu (ode1),
- metodu extrapolační (ode14x).

Metody, které jsou nabízeny pro simulaci systému s proměnným krokem [113]:

- diskrétní – obdoba pi řešení s konstantním vzorkovacím krokem,
- Darmond–Price (ode45),
- Bogacki–Shampine (ode23),
- Adamsova metoda (ode113),
- metoda numerické derivace (numerical differentiation formula – NDF, ode15s),

- Rosenbrock (ode23s),
- lichoběžníková metoda (ode23t),
- metoda TR-BDF2 (ode23tb).

Během výzkumu byla prakticky výhradně používána metoda řešení s pevným krokem (Fixed-Step) a diskrétním časem (discrete – no continuous states). Nastavení bylo voleno proto, že přechodné jevy nejsou zkoumány.

Pozn. Prověřit, zda simulace obsahuje nebo neobsahuje procesy s diskrétním, resp. kontinuálním, časem lze empiricky. Nejjednodušším způsobem je, zvolit jako metodu řešení metodu s diskrétním časem. Pokud je volba špatná, Simulink uživatele upozorní.

Simulační krok, resp. vzorkování (Fixed-step size, fundamental sample time), byl nastaven na 0,1. Pokud by byla ponechána výchozí hodnota, byl by simulační krok vypočítán jako rozdíl mezi počátečním a posledním krokem simulace, podělený 50. Pokud by simulace začínala krokem 1 a končila krokem 10, byl by krok nastaven na 0,2. Menší krok přináší větší přesnost simulace, ale prodlužuje čas výpočtu [113].

Na nastavení velikosti kroku je závislé nastavení dalších parametrů simulace. Vzorkovací frekvence simulace musí souhlasit s vzorkovací frekvencí pulzních generátorů, které de facto udávají takt simulace.

Ve vytvořeném modelu je třeba upravit původně nastavený simulační krok pouze tehdy, je-li požadována větší rychlost generování paketů. Při nastavení simulačního kroku na 0,1 je nejvyšší možná rychlost generování paketů v paketových generátorech 1 paket za 1 *s* simulačního času (resp. 10 vzorků). Pokud by bylo požadováno zvýšení rychlosti generování, např. na 1 paket za 0,5 *s*, bylo by třeba zmenšit krok na 0,01. Adekvátně by ale bylo třeba nastavit stejnou vzorkovací frekvenci ve všech blocích modelu, které požadují její pevné stanovení. Jsou to bloky:

- `Rychlost_generovani_paketu` – součást bloku `spoustec`,
- `Casovani_vystupu`,
- `Pulse_Generator` – součást bloku `Generator_paketu`, který je v modelu celkem 4×.

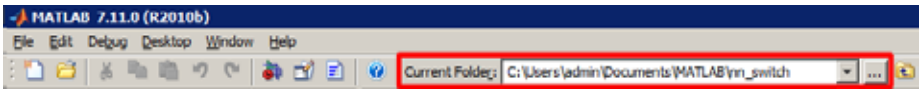
Při požadavku na snížení rychlosti generování paketů – např. na 1 paket za 2 s, není úprava vzorkovací frekvence nutná. Při běžné práci s modelem není nutná ani úprava rychlosti generování paketů. Při požadavku na vygenerování více paketů je jednodušší a transparentnější prodloužit dobu simulace modelu. Simulace v předkládané verzi začíná v čase 0 a končí v čase 20. Délka simulace je ale upravována v závislosti na prováděném experimentu.

V některých speciálních situacích je použit modul „Real-time“, tedy modifikující rychlost simulace tak, aby probíhala v pseudoreálném čase – tedy aby 1 *s* simulace odpovídala reálné 1 *s*. Speciálními situacemi se rozumí experimenty, u kterých je výhodné pozorovat výsledky simulace na ukazatelích, které se v průběhu simulace mění, případně časovou sousledost jevů atd. Modul „Real-time“ je postaven nad skriptem „waitforreal“, který publikoval Stan Quinn a který je prezentován MathWorks Inc. [114]. Skript a modul pro Simulink je možné použít i bez modulu „Real-Time Window Target“.

6 NNSWITCH – OVĚŘENÍ FUNKČNOSTI MODELU A DOSAŽENÝCH VÝSLEDKŮ

Kapitola popisuje základy ovládání vytvořeného modelu a postupy, jak je spustit a otestovat. Všechny vytvořené modely a jejich moduly byly vytvořeny v MATLABu 7.11.0 (R2010b).

Pracovní adresář MATLABu by měl být nastaven na adresář, ve kterém je uložen model. V modelu jsou odkazovány skripty (S-funkce), které by Simulink nenašel, viz obr. 6.1.



Obr. 6.1: Nastavení cesty, MATLAB 2010b

Pro spuštění simulace je potřeba MATLAB vybavený Simulinkem, knihovnami „Signal Processing Blockset“ a „Neural Network Toolbox“.

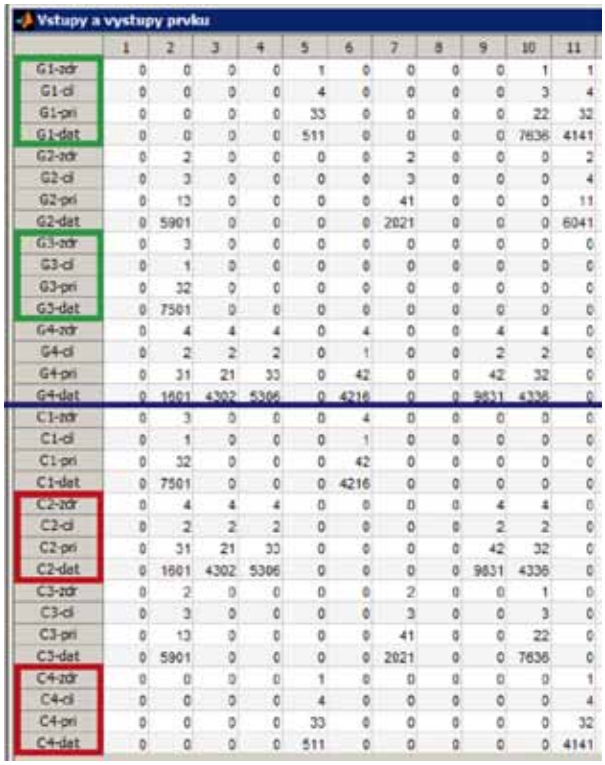
Základní a od něj odvozené modely jsou pojmenovány `nn_switch_prep_mat_v008.mdl`. Model je několikavrstvý – vrstvami lze procházet buď pomocí stromového navigačního panelu (Model Browser), nebo přímo otevření jednotlivých bloků. Některé subsystémy používají „masky“ – ty lze prohlédnout po klepnutí na „View Mask“ a „Look Under Mask“ v kontextové nabídce daného bloku.

Před spuštěním simulace je doporučeno vymazat „pracovní prostor MATLABu“ (`clear all`). Po provedení simulace (v Simulinku ctrl+t), lze v subsystému „`Monitoring`“ v 1. vrstvě a ve stejně pojmenovaném subsystému v bloku „`Krizovy_spinac_s_rizenim`“ sledovat dílčí výsledky simulace. Součty jednotlivých paketů lze sledovat i v rámci monitorovacího subsystému „Blok pocitadel paketu“. Zařazením bloku uloženého v souboru „`RT.mdl`“ lze celou simulaci zpomalit tak, aby bylo možné výstupy jednotlivých částí pohodlně sledovat. Popis bloku viz kap. 5.7.

Klíčový nástroj pro monitorování průběhu a výsledků je ale tvořen skriptem „`Vyhodnoceni.m`“. Skript byl vytvořen v několika modifikacích v návaznosti na sledovanou problematiku a model. Skript „`Vyhodnoceni_v3.m`“ provádí statistickou analýzu odeslaných a přijatých paketů. V simulinkové části modelu, k jehož analýze je skript použit, musí být vložen monitorovací subsystém „Blok pocitadel paketu“, příkladem takového modelu je `nn_switch_prep_mat_v008_srovnani_ex1.mdl`.

Skript je třeba spustit po ukončení simulace u MATLABu. Skript vygeneruje tabulku o 41 sloupcích zachycující vygenerované pakety a pakety zpracované – po průchodu přepínačem. Příklad takové tabulky je na obr. 6.2. Poslední 41. sloupec paketů je sice přepínačem vygenerován, ale není aktivním prvkem, ani počitadly dále zpracováván.

Na obr. 6.2 je zachycena pouze část tabulky. Tabulka je rozdělena na 2 části – část výstupů generátorů a část odchozích portů. Části jsou na obr. 6.2 pro přehlednost odděleny modrou čárou. Dále jsou zvýrazněny zeleně výstupy generátoru G1 a generátoru G3. Obdobně jsou červeně zvýrazněny řádky zachycující cílové porty C2 a C4. Jednotlivé sloupce odpovídají stavu generátorů/výstupních portu v daném kroku. Struktura řádků a popis tabulky odpovídá popisu v kap. 5.6, obr. 5.10.



Vstupy a vystupy prvku

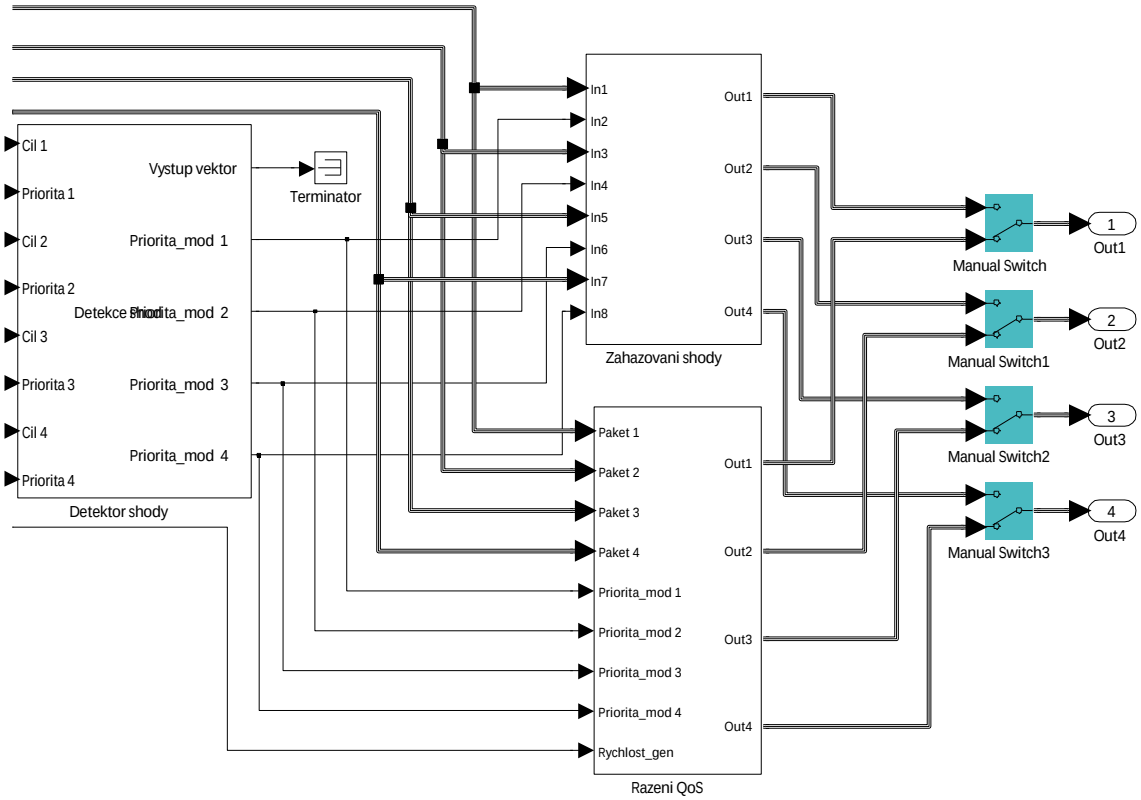
	1	2	3	4	5	6	7	8	9	10	11
G1-zdr	0	0	0	0	1	0	0	0	0	1	1
G1-cl	0	0	0	0	4	0	0	0	0	3	4
G1-pri	0	0	0	0	33	0	0	0	0	22	32
G1-dat	0	0	0	0	511	0	0	0	0	7636	4141
G2-zdr	0	2	0	0	0	0	2	0	0	0	2
G2-cl	0	3	0	0	0	0	3	0	0	0	4
G2-pri	0	13	0	0	0	0	41	0	0	0	11
G2-dat	0	5901	0	0	0	0	2021	0	0	0	6041
G3-zdr	0	3	0	0	0	0	0	0	0	0	0
G3-cl	0	1	0	0	0	0	0	0	0	0	0
G3-pri	0	32	0	0	0	0	0	0	0	0	0
G3-dat	0	7501	0	0	0	0	0	0	0	0	0
G4-zdr	0	4	4	4	0	4	0	0	4	4	0
G4-cl	0	2	2	2	0	1	0	0	2	2	0
G4-pri	0	31	21	33	0	42	0	0	42	32	0
G4-dat	0	1601	4302	5306	0	4216	0	0	9831	4336	0
C1-zdr	0	3	0	0	0	4	0	0	0	0	0
C1-cl	0	1	0	0	0	1	0	0	0	0	0
C1-pri	0	32	0	0	0	42	0	0	0	0	0
C1-dat	0	7501	0	0	0	4216	0	0	0	0	0
C2-zdr	0	4	4	4	0	0	0	0	4	4	0
C2-cl	0	2	2	2	0	0	0	0	2	2	0
C2-pri	0	31	21	33	0	0	0	0	42	32	0
C2-dat	0	1601	4302	5306	0	0	0	0	9831	4336	0
C3-zdr	0	2	0	0	0	0	2	0	0	1	0
C3-cl	0	3	0	0	0	0	3	0	0	3	0
C3-pri	0	13	0	0	0	0	41	0	0	22	0
C3-dat	0	5901	0	0	0	0	2021	0	0	7636	0
C4-zdr	0	0	0	0	1	0	0	0	0	0	1
C4-cl	0	0	0	0	4	0	0	0	0	0	4
C4-pri	0	0	0	0	33	0	0	0	0	0	32
C4-dat	0	0	0	0	511	0	0	0	0	0	4141

Obr. 6.2: Příklad vyhodnocovací tabulky

Chování modelu lze měnit zapojováním interních paralelně pracujících bloků. Příklad takového zapojení je v bloku vstupních front (`Vst_fronty`). Výřez bloku je zachycen na obr. 6.3. Na obr. 6.3 jsou patrné 2 paralelně pracující bloky „`Zahazovani shody`“ a „`Razeni QoS`“. Který z bloků bude zaslán do spojovacího pole, je rozlišeno přepnutím manuálního přepínače – na obr. 6.3 je označen „`Manual switch`“ a je podbarven zeleně.

Měnit nastavení lze individuálně pro každý ze vstupních portů. Obdobné manuální přepínače jsou součástí každého z generátorů, kde umožňují volit mezi náhodným generováním buď celého obsahu paketu, nebo jeho jednotlivými poli. Podobně jsou manuálně konfigurovatelné periody generátorů v bloku „`Spoustec_g`“.

Pokud jsou přepínači jako výstupy zvoleny předdefinované sekvence, lze je individuálně definovat v rámci jednotlivých bloků jako vektory – příkladem takových bloků jsou bloky „`Cilovy port_stat`“, „`Priorita_stat`“ a „`Predn_data_stat`“, které jsou součástí paketových generátorů.



Obr. 6.3: Výřez zapojení bloku vstupních front

ZÁVĚR

V rámci řešení disertační práce byl navržen nový aktivní prvek – přepínač řízený neuronovou sítí. Všechny cíle definované v kapitole 2 se podařilo v plném rozsahu naplnit. V první fázi řešení disertační práce proběhl výzkum v současnosti dostupných technologií a aktuálního stavu poznání v oboru přepínaných sítí a konstrukci síťových prvků. Pro vývoj nového způsobu řízení byly zvažovány 2 síťové prvky – přepínače a směrovače. Vzhledem k postupnému prolínání byl zvolen k návrhu aktivní prvek – přepínač. Koncepčně je cílen do rolí prvku na hraně operátora (PE). Přepínač je navržen tak, aby respektoval soudobé požadavky na QoS.

Prvním vytčeným cílem disertační práce byla analýza a měření dostupných aktivních prvků. Výsledky měření jsou shrnuty v kap. 3.1. Dílčí měření jsou také obsažena v kap. 1.3.1 a 4.1. Byly proměřeny vlastnosti přepínačů Cisco, Hewlett-Packard, Micronet a platformy VNUML. Platforma VNUML byla do testů zařazena zejména proto, že byla v úvodu výzkumu problematiky řešené v rámci disertační práce zvažována jako perspektivní pro testovací implementaci navrženého řešení. Postupně se ale platforma VNUML ukázala jako nevhodnou volbou. Problematikou návrhu modelu v rámci VNUML se zabývá kap. 4.1.

V rámci návrhu nového přepínače a zejména jeho řízení bylo třeba zvolit neuronovu síť, která by byla pro řízení aktivního prvku nejvhodnější. O této problematice pojednává kap. 3.3. Jednalo se o druhý z cílů disertační práce. V návaznosti na předchozí výzkum např. [86] a [87] byly zvažovány perceptronové umělé neuronové sítě a sítě „feedforward backpropagation“ (dopředné šíření vstupního signálu tréninkového vzoru a zpětným šířením chyby). Po provedení komparačních testů (kap. 3.3.3) byla pro řízení navrhovaného přepínače zvolena neuronová síť „feedforward backpropagation“. V rámci komparačních testů byly neurovou sítí řešeny problémy logických funkcí antivalence a disjunkce. V případě řešení disjunkce obstály oba srovnávané typy neuronové sítě. Řešit problém antivalence, která je charakterem podobná problému rozhodování o cíli paketu, byla schopna jen síť „feedforward backpropagation“. Proto byla pro další návrhy zvolena právě ta.

Dalším cílem disertační práce bylo navrhnout koncepčně nový přepínač. Touto problematikou se zabývá jak část kap. 3, tak kap. 4. Byly analyzovány typy spojovacích polí – kvůli možnosti externího řízení a kontrolovatelnosti bylo rozhodnuto o použití křížového spínače (kap. 3.2.1). Byla také navrhována celková koncepce aktivního prvku – bylo rozhodnuto o návrhu čtyřportového přepínače. Bylo také rozhodnuto o umístění softwarových front. Oproti běžně používanému umístění za spojovací pole před hardwarové fronty byly softwarové fronty situovány naopak před spojovací pole. Toto umístění má řadu nevýhod – např. problémy blokování paketů uvnitř fronty, viz kap. 3.2.2. Z hlediska navrhovaného modelu se ale ukázalo zvolené řešení jako nejlepší. Pro správu paketů ve frontách byl navržen algoritmus Modified Deficit Round-Robin (modifikovaný deficitní okruh), který byl upraven do podoby striktní priority (viz kap. 3.2.2). Při analýze algoritmů pro správu softwarových front byla zkoumána i možnost jejich správy neuronovou sítí. Postupně se ale ukázalo použití neuronové sítě jako komplikovanější a výsledky hůře predikovatelné než v případě použití konvenčních algoritmů. Proto bylo řízení správy softwarových front ponecháno na zmíněném konvenčním algoritmu.

Protože modelování celého RM ISO/OSI, resp. TCP/IP se ukázalo být neefektivní, byl jako jeden z cílů disertační práce stanoven návrh výrazně zjednodušeného protokolu vhodného pro řešení stanovené problematiky. Byl navržen protokol obsahující nejvýznamnější pole se vztahem ke QoS a směrováním paketu v rámci aktivního prvku. Navržený protokol má v hlavičce pouze 3 pole, datová část je čtvrtá. V hlavičce jsou obsažena pouze pole adres (zdroj

a cíl paketu) a pole priority, které je inspirováno polem DS, resp. DSCP IP protokolu. Datová část slouží také k identifikaci paketu při kontrole práce navrženého přepínače.

Posledním z cílů disertační práce bylo všechny navržené koncepce ověřit v rámci komplexního simulačního modelu. Vzhledem k tomu, že výzkum, resp. řešení disertační práce, probíhal nelineárně, v prvotních fázích bylo uvažováno spíše o modelování v rámci VNUML. Později byly všechny modely tvořeny pouze v rámci MATLABu a Simulinku. Bylo vytvořeno několik dílčích simulačních modelů a několik verzí komplexního modelu. Modelováním a výsledky se zabývá zejména kap. 5. Postupnými úpravami modelu, simulačních časů, hustoty generování paketů atd., se podařilo dosáhnout nulové ztrátovosti paketů v navrženém přepínači. Navržený přepínač a jeho model potvrdily, že síťový prvek lze neuronovu sítí řídit.

Pomocí provedených simulací se podařilo model optimalizovat a připravit jej pro hardwarovou implementaci. Na výzkum provedený v rámci disertační práce by v budoucnu měla navázat implementace do vývojových karet INVEA-TECH Combo-20G, tedy použití v reálném síťovém prostředí.

Dílčí výsledky byly průběžně prezentovány v rozličných publikacích. Všechny vytčené cíle disertační práce byly dosaženy.

SEZNAM POUŽITÝCH ZDROJŮ

[1] PETERKA, Jiří. Báječný svět počítačových sítí – ve znamení konvergence. *PC World.* 2005, Sv. 3.

[2] Cisco Systems, Inc. Global - 2015 Forecast Highlights. *VNI Forecast Highlights.* [Online] [Citace: 15. 8 2014.] http://www.cisco.com/web/solutions/sp/vni/vni_forecast_highlights/index.html.

[3] DOSTÁLEK, Libor a KABELOVÁ, Alena. *Velký průvodce protokolyTCP/IP a systémem DNS.* Praha : Computer Press, 2000. ISBN: 80-7226-323-4.

[4] International Organization for Standardization. ISO/IEC 7498: Information technology – Open Systems Interconnection – Basic Reference Model: The Basic Model. Genève : International Organization for Standardization, 1994.

[5] LAW, David J., a další. IEEE 802.3-2012: IEEE Standard for Ethernet. [Online] 28. 12 2012. [Citace: 10. 3 2014.] http://standards.ieee.org/about/get/802/802.3.html. ISBN 973-07381-7312-2.

[6] GEBALI, Fayez. *Analysis of Computer and Communication Networks.* New York : Springer Science+Business Media, LLC, 2008. ISBN 978-0-387-74436-0.

[7] WALLACE, Kevin. *Cisco VoIP.* Brno : Computer Press, a. s., 2009. ISBN 978-80-251-2228-0.

[8] CASNER, S. a JACOBSON, V. *RFC 2508: Compressing IP/UDP/RTP Headers for Low-Speed Serial Links.* [Online] 2 1999. [Citace: 9. 11 2010.] http://datatracker.ietf.org/doc/rfc2508/.

[9] KOREN, T., CASNER, S. a BORMANN, C. *RFC 3544: IP Header Compression over PPP.* [Online] 7 2003. [Citace: 15. 8 2011.] http://datatracker.ietf.org/doc/rfc3544/.

[10] KOREN, T., a další. *RFC 3545: Enhanced Compressed RTP (CRTP) for Links with High Delay, Packet Loss and Reordering.* [Online] 7 2003. [Citace: 11. 8 2011.] http://datatracker.ietf.org/doc/rfc3545/.

[11] SCHULZRINNE, H., a další. *RFC 3550: RTP: A Transport Protocol for Real-Time Applications.* [Online] 7 2003. [Citace: 3. 5 2011.] http://datatracker.ietf.org/doc/rfc3550/.

[12] JOHANSSON, I. a WESTERLUND, M. *RFC 5506: Support for Reduced-Size Real-Time Transport Control Protocol (RTCP): Opportunities and Consequences.* [Online] 4 2009. [Citace: 8. 1 2011.] http://datatracker.ietf.org/doc/rfc5506/.

[13] PERKINS, C. a WESTERLUND, M. *RFC 5761: Multiplexing RTP Data and Control Packets on a Single Port.* [Online] 4 2010. [Citace: 4. 5 2011.] http://datatracker.ietf.org/doc/rfc5761/. ISSN: 2070-1721.

[14] PERKINS, C. a SCHIERL, T. *RFC 6051: Rapid Synchronisation of RTP Flows.* [Online] 11 2010. [Citace: 25. 3 2011.] http://datatracker.ietf.org/doc/rfc6051/. ISSN 2070-1721.

[15] BEGEN, A., PERKINS, C. a WING, D. *RFC 6222: Guidelines for Choosing RTP Control Protocol (RTCP) Canonical Names (CNAMEs).* [Online] 4 2011. [Citace: 25. 7 2011.] http://datatracker.ietf.org/doc/rfc6222/. ISSN: 2070-1721.

[16] International Telecommunication Union. Series G: Transmission Systems and Media, Digital Systems and Networks. *G.114: One-way transmission time.* [Online] 5. 2003. http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.114-200305-I!!PDF-E&type=items.

[17] DEMICHELIS, C. a CHIMENTO, P. *RFC 3393: IP Packet Delay Variation Metric for IP Performance Metrics (IPPM).* [Online] 11 2002. [Citace: 15. 4 2010.] http://datatracker.ietf.org/doc/rfc3393/.

[18] Cisco Systems, Inc. Cisco 3900 Series Integrated Services Routers. [Online] [Citace: 10. 8 2011.] http://www.cisco.com/en/US/prod/collateral/routers/ps10536/data_sheet_c78_553924.html.

[19] SCHUDEL, Gregg. Bandwidth, Packets Per Second, and Other Network Performance Metrics. *Cisco Security Intelligence Operations.* [Online] [Citace: 20. 1 2014.] http://www.cisco.com/web/about/security/intelligence/network_performance_metrics.html.

[20] Cisco Systems, Inc. Cisco Catalyst 4500 Series Switches. [Online] [Citace: 10. 8 2011.] http://www.cisco.com/en/US/products/hw/switches/ps4324/prod_models_comparison.html.

[21] ODOM, Wendell, HEALY, Rus a MEHTA, Naren. *Směrování a přepínání sítí – Autorizovaný výukový průvodce.* Brno : Computer Press, a.s.; Cisco System, Inc., 2009. ISBN: 978-80-251-2520-5.

[22] ANDERSSON, L. a MADSEN, T. *RFC 4026: Provider Provisioned Virtual Private Network (VPN) Terminology.* [Online] 2005 5. [Citace: 29. 6 2013.] http://datatracker.ietf.org/doc/rfc4026/.

[23] GROW, Robert M., a další. IEEE 802.3-2008: Ethernet. [Online] 26. 12 2008. [Citace: 12. 6 2011.] http://standards.ieee.org/about/get/802/802.3.html. ISBN 973-07381-5796-2.

[24] JEFFREE, Tony a SEAMAN, Mick. IEEE 802.1Q-2005: IEEE Standard for Local and metropolitan area networks, Virtual Bridged Local Area Networks. [Online] 19. 5 2006. [Citace: 16. 8 2011.] http://standards.ieee.org/getieee802/download/802.1Q-2005.pdf. ISBN 0-7381-4877-6.

[25] Cisco Systems, Inc. Inter-Switch Link and IEEE 802.1Q Frame Format. [Online] 25. 8 2006. [Citace: 10. 8 2011.] http://www.cisco.com/en/US/tech/tk389/tk689/technologies_tech_note09186a0080094665.shtml.

[26] —. Configuring QoS. *Catalyst 2960 Switch Software Configuration Guide.* [Online] 8 2007. [Citace: 18. 8 2011.] http://www.cisco.com/en/US/docs/switches/lan/catalyst2960/software/release/12.2_37_ey/configuration/guide/swqos.html.

[27] Information Sciences Institute, University of Southern California. *RFC 791: Internet Protocol Darpa Internet Program Protocol Specification.* [Online] 9 1981. [Citace: 23. 6 2011.] http://datatracker.ietf.org/doc/rfc791/.

[28] ALMQUIST, P. *RFC 1349: Type of Service in the Internet Protocol Suite.* [Online] 7 1992. [Citace: 3. 6 2011.] http://datatracker.ietf.org/doc/rfc1349/.

[29] NICHOLS, K., a další. *RFC 2474: Definition of the Differentiated Services Field (DS Field) in the IPv4 and IPv6 Headers.* [Online] 12 1998. [Citace: 18. 8 2011.] http://datatracker.ietf.org/doc/rfc2474/.

[30] JACOBSON, V., NICHOLS, K. a PODURI, K. *RFC 2598: An Expedited Forwarding PHB.* [Online] 6 1999. [Citace: 10. 8 2011.] http://datatracker.ietf.org/doc/rfc2598/.

[31] DAVIE, B., a další. *RFC 3246: An Expedited Forwarding PHB (Per-Hop Behavior).* [Online] 3 2002. [Citace: 10. 8 2011.] http://datatracker.ietf.org/doc/rfc3246/.

[32] RAMAKRISHNAN, K., FLOYD, S. a BLACK, D. *RFC 3168: The Addition of Explicit Congestion Notification (ECN) to IP.* [Online] 9 2001. [Citace: 18. 8 2011.] http://datatracker.ietf.org/doc/rfc3168/.

[33] KENT, S. a SEO, K. *RFC 4301: Security Architecture for the Internet Protocol.* [Online] 12 2005. [Citace: 18. 8 2011.] http://datatracker.ietf.org/doc/rfc4301/.

[34] BRISCOE, B. *RFC 6040: Tunnelling of Explicit Congestion Notification.* [Online] 11 2010. [Citace: 18. 8 2011.] http://datatracker.ietf.org/doc/rfc6040/. ISSN: 2070-1721.

[35] BRADNER, S. a PAXSON, V. *RFC 2780: IANA Allocation Guidelines For Values In the Internet Protocol and Related Headers.* [Online] 3 2000. [Citace: 10. 8 2011.] http://datatracker.ietf.org/doc/rfc2780/.

[36] BLAKE, S., a další. *RFC 2475: An Architecture for Differentiated Services.* [Online] 12 1998. [Citace: 19. 8 2011.] http://datatracker.ietf.org/doc/rfc2475/.

[37] HEINANEN, J., a další. *RFC 2597: Assured Forwarding PHB Group.* [Online] 6 1999. [Citace: 19. 8 2011.] http://datatracker.ietf.org/doc/rfc2597/.

[38] CHAO, Jonathan H. a GUO, Xiaolei. *Quality of Service Control in High-Speed Networks.* New York : John Wiley & Sons, Inc., 2002. ISBN 0-471-00397-2.

[39] BABIARZ, J., CHAN, K. a BAKER, F. *RFC 4594: Configuration Guidelines for DiffServ Service Classes.* [Online] 8 2006. [Citace: 3. 9 2011.] http://datatracker.ietf.org/doc/rfc4594/.

[40] DEERING, S. a HINDEN, R. *RFC 2460: Internet Protocol, Version 6 (IPv6) Specification.* [Online] 12 1998. [Citace: 5. 6 2011.] http://datatracker.ietf.org/doc/rfc2460/.

[41] —. *RFC 1883: Internet Protocol, Version 6 (IPv6) Specification.* [Online] 12 1995. [Citace: 3. 6 2011.] http://datatracker.ietf.org/doc/rfc1883/.

[42] SATRAPA, Pavel. *IPv6 – Internet Protokol verze 6.* Praha : CZ.NIC, z. s. p. o., 2008. ISBN: 978-80904248-0-7.

[43] RAJAHALME, J., a další. *RFC 3697: IPv6 Flow Label Specification.* [Online] 3 2004. [Citace: 2. 6 2011.] http://datatracker.ietf.org/doc/rfc3697/.

[44] ROSEN, E., VISWANATHAN, A. a CALLON, R. *RFC 3031: Multiprotocol Label Switching Architecture.* [Online] 1 2001. [Citace: 15. 6 2011.] http://datatracker.ietf.org/doc/rfc3031/.

[45] ROSEN, E., a další. *RFC 3032: MPLS Label Stack Encoding.* [Online] 1 2001. [Citace: 8. 6 2011.] http://datatracker.ietf.org/doc/rfc3032/.

[46] ANDERSSON, L. a ASATI, R. *RFC 5462: Multiprotocol Label Switching (MPLS) Label Stack Entry: "EXP" Field Renamed to "Traffic Class" Field.* [Online] 2 2009. [Citace: 15. 6 2011.] http://datatracker.ietf.org/doc/rfc5462/.

[47] LE FAUCHEUR, F., a další. *RFC 3270: Multi-Protocol Label Switching (MPLS) Support of Differentiated Services.* [Online] 5 2002. [Citace: 18. 6 2011.] http://datatracker.ietf.org/doc/rfc3270/.

[48] DAVIE, B., BRISCOE, B. a TAY, J. *RFC 5129: Explicit Congestion Marking in MPLS.* [Online] 1 2008. [Citace: 15. 6 2011.] http://datatracker.ietf.org/doc/rfc5129/.

[49] Cisco Systems, Inc. Network Based Application Recognition (NBAR). [Online] [Citace: 2. 3 2014.] http://www.cisco.com/c/en/us/products/ios-nx-os-software/network-based-application-recognition-nbar/index.html.

[50] JEFFREE, Tony, a další. IEEE 802.1AE-2006: Media Access Control (MAC) Security. [Online] 18. 8 2006. [Citace: 1. 5 2011.] http://standards.ieee.org/getieee802/download/802.1AE-2006.pdf. ISBN 0-7381-4991-8.

[51] JEFFREE, Anthony, CONGDON, Paul a HADDOCK, Stephen. IEEE 802.1Q-2011: IEEE Standard for Local and metropolitan area networks – Media Access Control (MAC) Bridges and Virtual Bridge Local Area Networks. [Online] 31. 8 2011. [Citace: 5. 10 2013.] http://standards.ieee.org/getieee802/download/802.1Q-2011.pdf. ISBN 978-0-7381-6708-4.

[52] BAKER, F., a další. *RFC 5865: A Differentiated Services Code Point (DSCP) for Capacity-Admitted Traffic.* [Online] 5 2010. [Citace: 15. 1 2014.] http://datatracker.ietf.org/doc/rfc5865/.

[53] Cisco Systems Inc. Understanding and Tuning the tx-ring-limit Value. *IP to ATM Class of Service.* [Online] 14. 12 2007. [Citace: 27. 11 2011.] http://www.cisco.com/en/US/tech/tk39/tk824/technologies_tech_note09186a00800fbafc.shtml.

[54] Cisco Systems, Inc. *Cisco 7200 Series Design Library: ATM Traffic Management.* San Jose : Cisco Systems, Inc., 2005. OL-3274-01.

[55] CHAO, Jonathan H. a LIU, Bin. *High Performance Switches and Routers.* New Jersey : JohnWiley & Sons, Inc., 2007. ISBN 13: 978-0-470-05367-6.

[56] ODOM, Wendell a CAVANAUGH, Michael J. *Cisco QoS Exam Certification Guide (IP Telephony Self-Study), 2nd Edition.* Indianapolis : Cisco Press, 2004. 1-58720-124-0.

[57] Cisco Systems, Inc. Congestion Management Overview. *QoS: Congestion Management Configuration Guide, Cisco IOS Release 15M&T.* [Online] 4. 3 2013. [Citace: 6. 4 2014.] http://www.cisco.com/c/en/us/td/docs/ios-xml/ios/qos_conmgt/configuration/15-mt/qos-conmgt-15-mt-book.html.

[58] Juniper Networks, Inc. Configuring Excess Bandwidth Sharing. [Online] 9. 5 2014. [Citace: 15. 6 2014.] http://www.juniper.net/techpubs/en_US/junos14.1/topics/usage-guidelines/cos-configuring-excess-bandwidth-sharing.html.

[59] Brocade Communications Systems, Inc. Traffic class scheduling policy. *Network OS Administrator's Guide.* [Online] 2014. [Citace: 29. 6 2014.] http://www.brocade.com/downloads/documents/html_product_manuals/NOS_411_AG_04/GUID-26A38E50-6CCD-4EFE-826C-D07378DC970F.html.

[60] PILAŘ, Jaromír. *Přednáška Catalyst Instant Access (konference Cisco Connect).* Praha : autor neznámý, 2014.

[61] LEKKAS, Panos C. *Network Processors – Architectures, Protocols and Platforms.* New York : McGraw-Hill, 2003. ISBN 0-07-140986-6.

[62] BONG-CHEOL, Kim, SUN-GI, Kim a YONG-SEOK, Park. *Queue sharing with fair rate guarantee. US7792131* USA, 26. 7 2007. Queuing arrangement.

[63] BOCH, Radek. *Přednáška Chytrá přístupová vrstva v LAN sítích (konference Cisco Expo).* Praha : autor neznámý, 2011.

[64] Cisco Systems, Inc. Understand and Configure MDRR/WRED on the Cisco 12000 Series Internet Router. *Cisco 12000 Series Routers.* [Online] 6. 3 2008. [Citace: 6. 10 2009.] http://www.cisco.com/c/en/us/support/docs/routers/12000-series-routers/18841-mdrr-wred-18841.html.

[65] POSTEL, Jon. *RFC 793: Transmission Control Protocol.* [Online] 9 1981. [Citace: 6. 7 2012.] http://datatracker.ietf.org/doc/rfc793/.

[66] ALLMAN, M., PAXSON, V. a BLANTON, E. *RFC 5681: TCP Congestion Control.* [Online] 9 2009. [Citace: 4. 8 2013.] http://datatracker.ietf.org/doc/rfc5681/.

[67] FLOYD, Sally a JACOBSON, Van. Random Early Detection Gateways for Congestion Avoidance. *IEEE/ACM Transactions on Networking.* 8 1993, Sv. 1, 4, stránky 397-413.

[68] DEMEL, Jiří. Markovské řetězce. [autor knihy] Jiří Demel. *Operační výzkum.* Praha : Katedra Inženýrské Informatiky, ČVUT, 2011.

[69] NAVARA, Mirko. Markovovy řetězce. [autor knihy] Mirko Navara. *Pravděpodobnost, statistika a teorie informace.* Praha : Katedra kybernetiky FEL ČVUT, 2013.

[70] United Business Media Limited. Switch-Fabric Chipsets. *Light Reading – Networking the Telecom Industry.* [Online] United Business Media Limited, 3. 3 2004. [Citace: 12. 2 2009.] http://www.lightreading.com/document.asp?page_number=1&site=&doc_id=47959.

[71] LAWSON, Stephen. Network processors enter new generation. *Network World.* [Online] Network World, Inc., 19. únor 2006. [Citace: 24. duben 2009.] http://www.networkworld.com/news/2002/0619processors.html.

[72] ELHANANY, Itamar and HAMDI, Mounir. *High-performance Packet Switching Architectures.* London : Springer, 2007. ISBN: 1-84628-273-X.

[73] WEIDONG, Wu. *Packet forwarding technologies.* New York : Taylor & Francis Group, LLC, 2008. ISBN: 978-0-8493-8057-0.

[74] SEIFERT, Rich a EDWARDS, Jim. *The All-New Switch Book – The Complete Guide to LAN Switching Technology.* Indianapolis : Wiley Publishing, Inc., 2008. ISBN: 978-0-470-28715-6.

[75] MEDHI, Deepankar a RAMASAMY, Karthikeyan. *Network Routing – Algorithms, Protocols, and Architectures.* San Francisco : Morgan Kaufmann Publishers, Elsevier Inc., 2007. ISBN13: 978-0-12-088588-6.

[76] GILADI, Ran. *Network Processors – Architecture, Programming, and Implementation.* Burlington : Elsevier Inc. Morgan Kaufmann Publishers, 2008. ISBN 978-0-12-370891-5.

[77] SOLDER, Carl. Cisco Catalyst 6500 Supervisor 2T Architecture. [Online] 2011. [Citace: 22. 6 2011.] http://www.cisco.com/en/US/prod/collateral/switches/ps5718/ps708/white_paper_c11-676346.pdf.

[78] CROWLEY, Patrick, a další. *Network Processor Design, Issues and Practices, Volume 2.* San Francisco : Morgan Kaufmann Publishers, 2004. ISBN: 0-12198-157-6.

[79] VYNCKE, Eric a PAGGEN, Christopher. *LAN Switch Security, What Hackers Know About Your Switches.* Indianapolis : Cisco Press, 2008. ISBN 978-1-58705-256-9.

[80] CHAO, Jonathan H., JING, Zhigang a LIEW, Soung Y. Matching Algorithms for Three-Stage Bufferless Clos Network Switches. *IEEE Communications Magazine.* 10 2003.

[81] OKI, Eiji, a další. Concurrent Round-Robin-Based Dispatching Schemes for Clos-Network Switches. *IEEE/ACM Transactions on Networking.* 12 2002, Sv. 10.

[82] Cisco Systems, Inc. Catalyst Switched Port Analyzer (SPAN) Configuration Example. [Online] 16. 7 2007. [Citace: 2. 3 2010.] http://www.cisco.com/en/US/products/hw/switches/ps708/products_tech_note09186a008015c612.shtml.

[83] *Wire-speed Traffic Management in Ethernet Switches.* MISHRA, Shridhar Mubaraq, a další. místo neznámé : IEEE, 2003. Sv. II, stránky 105-108. 0-7803-7761-310.

[84] *A Self-routing Switch Fabric Architecture on a Chip.* JANG, Ho-Rang a KIM, Hyong S. místo neznámé : IEEE, 2008. stránky 1–5. 978-1-4244-2324-8/08.

[85] ŠKORPIL, Vladislav a ŠŤASTNÝ, Jiří. Alternatives of Converged Network Control. *Elektrorevue.* 2010, Sv. 1, 1, stránky 19–23.

[86] *Network Elements Controlled by Artificial Neural Network.* ŠKORPIL, Vladislav a NOVÁK, David. Corfu Island, Greece : WSEAS, 2010. Proceedings of 14th WSEAS International Conference on Communications. stránky 145–148. ISBN 978-960-474-200-4.

[87] *Back Propagation and Genetic Algorithms for Control of the Network Element.* ŠKORPIL, Vladislav a KAMBA, Stanislav. Budapest, Hungary : IEEE, 2011. 34th International Conference on Telecommunications and Signal Processing. stránky 240–243. ISBN 978-1-4577-1410-8.

[88] *A Neural Network Solution to QoS-IP Team-Optimal Dynamic Routing.* BAGLIETTO, M., a další. Seville : IEEE, 2005. 44th IEEE Conference on Decision and Control, and the European Control Conference 2005. stránky 7452–7459. 0-7803-9568-9/05/.

[89] JIANG, Michael Y. Neural Networks Demo using Matlab 6.5. [Online] 2001. http://www.dti.unimi.it/~ceselli/cro_2009/NNdemo_matlab.pdf.

[90] BEALE, Mark Hudson, HAGAN, Martin T. a DEMUTH, Howard B. *Neural Network Toolbox 7.* Natic : The MathWorks, Inc., 2010.

[91] *What Size Neural Network Gives Optimal Generalization? Convergence Properties of Backpropagation.* LAWRENCE, Steve, GILES, Lee C. a TSOI, Chung Ah. Washington, D.C. : Institute for Advanced Computer Studies University of Maryland, 1996.

[92] *Advanced Network Simulation under User-Mode Linux.* STEFFEN, Andreas, MARCHIONNI, Eric a RAYO, Patrik. Heute schon das Morgen sehen – 19. DFN-Arbeitstagung über Kommunikationsnetze, Düsseldorf. stránky 321-333.

[93] FERNÁNDEZ, David, MIGUEL, Tomás a GALÁN, Fermín. Study and Emulation of IPv6 Internet-Exchange-Based Addressing Models. *IEEE Communications Magazine.* 1 2004, stránky 105-112.

[94] DIKE, Jeff. *User Mode Linux.* místo neznámé : Prentice Hall, 2006. ISBN 0-13-186505-6.

[95] *Use of virtualization tools in computer network laboratories.* GALÁN, Fermín, a další. Istanbul : autor neznámý, 2004. Fifth International Conference on Information Technology Based Higher Education and Training (IEEE Cat. No.04EX898). str. 209. ISBN 0-7803-8596-9 .

[96] *Quantifying Performance Properties of Virtual Machine.* XU, Xianghua, ZHOU, Feng a JIANG, Jian Wan Yucheng. Shanghai : autor neznámý, 2008. International Symposium on Information Science and Engineering. str. 24. ISBN 978-1-4244-2727-4.

[97] BRANDNER, Scott a MCQUAID, J. *RFC 2544: Benchmarking Methodology for Network Interconnect Devices.* [Online] 3 1999. [Citace: 3. 9 2010.] http://datatracker.ietf.org/doc/rfc2544/.

[98] International Telecommunication Union. ITU-T Y.1564: Ethernet service activation test methodology. *Series Y: Global information infrastructure, Internet protocol aspects and next-generation networks.* [Online] 3 2011. [Citace: 5. 10 2013.] http://www.itu.int/rec/T-REC-Y.1564-201103-I/en.

[99] BRADNER, Scott. *RFC 1242: Benchmarking Terminology for Network Interconnection Devices.* [Online] 7 1991. [Citace: 3. 4 2011.] http://datatracker.ietf.org/doc/rfc1242/.

[100] Hewlett-Packard Development Company, L.P. *HP E2600 Switch Series - Specifications and Warranty.* [Online] 12 2010. http://h10010.www1.hp.com/wwpc/us/en/sm/WF06a/12883-12883-4172267-4172301-4172277-329892.html.

[101] BISHOP, Christopher M. *Neural Networks for Pattern Recognition.* Oxford : Oxford University Press, 1996. ISBN 978-0198538646.

[102] VOLNÁ, Eva. *Neuronové sítě 1.* Ostrava : Ostravská univerzita, Přírodovědecká fakulta, 2002.

[103] IBM Research. Brain Power. *IBM Research.* [Online] IBM Corporation, 2014. [Citace: 9. 8 2014.] http://www.research.ibm.com/cognitive-computing/neurosynaptic-chips.shtml#fbid=AVrCJIT1X4O.

[104] MATIGNON, Randall. *Neural Network Modeling using SAS Enterprise Miner.* 2005. ISBN 1-4184-6675-1.

[105] BARTSCH, Hans Jochen. *Matematické vzorce.* Praha : Mladá fronta, 2002. ISBN 80-204-0607-7.

[106] MARSHALL, Brad. *Virtual Machines under Linux.* SAGE-AU. Queensland : s.n., 2003. Presentation.

[107] *Virtual Network User-Mode-Linux (VNUML).* [Online] http://www.dit.upm.es/vnumlwiki/index.php/Main_Page.

[108] *Virtual Networks over Linux (VNX).* [Online] [Citace: 10. 7 2014.] http://www.dit.upm.es/vnx.

[109] POLÍVKA, Michal a PELKA, Tomáš. Simulátor rozsáhlých sítí – SimPP. *Elektrorevue - Internetový časopis.* 13. 12 2010, Sv. 117, stránky 1–5.

[110] *SimPP – Wide Area Network Simulator.* PELKA, Tomáš, POLÍVKA, Michal a HAJNÝ, Jan. Budapest : Brno University of Technology, 2010. The 33rd International Conference on Telecommunication and Signal Processing, TSP 2010. stránky 1–4. ISBN 978-963-88981-0-4.

[111] KACÁLEK, Jan a MÍČA, Ivan. Nelineární analýza a predikce sítového provozu. *Elektrorevue.* [Online] 27. 12 2009. [Citace: 1. 6 2012.] http://www.elektrorevue.cz/. ISSN 1213-1539.

[112] WIDE MAWI WorkingGroup. Packet traces from WIDE backbone. *MAWI Working Group Traffic Archive.* [Online] WIDE MAWI WorkingGroup. [Citace: 6. 8 2010.] http://mawi.wide.ad.jp/mawi/.

[113] The MathWorks, Inc. *Simulink 7.* Natick : The MathWorks, Inc., 2011.

[114] —. *How do I get a real time clock in simulations using Simulink?* [Online] 7 2010. [Citace: 15. 1 2012.] http://www.mathworks.com/support/solutions/en/data/1-15JAW/.

[115] POSTEL, J. *RFC 768: User Datagram Protocol.* [Online] 28. 8 1980. [Citace: 15. 5 2013.] http://datatracker.ietf.org/doc/rfc768/.

[116] FRYSINGER, James R., a další. IEEE 1541-2002: IEEE Standard for Prefixes for Binary Multiples. [Online] 13. 2 2003. [Citace: 22. 7 2011.] http://standards.ieee.org/findstds/standard/1541-2002.html. ISBN 0-7381-3386-8.

[117] FRYSINGER, James R., a další. IEEE 260.1-2004: IEEE Standard Letter Symbols for Units of Measurement. [Online] 24. 9 2004. [Citace: 5. 6 2011.] http://standards.ieee.org/findstds/standard/260.1-2004.html. ISBN 0-7381-3998-X.

[118] FROOM, Richard, FLANNAGAN, Mike a TUREK, Kevin. *Cisco Catalyst QoS: Quality of Service in Campus Networks.* Indianapolis : Cisco Press, 2003. ISBN-13: 978-1-58705-120-3.

SEZNAM ZKRATEK

Zkratka	Popis
ACK	ACKnowledgement
ACL	Access Control List
AF	Assured Forwarding – zaručené doručování
ASIC	Application-Specific Integrated Circuit
ATM	Asynchronous Transfer Mode
CAM	Content Addressable Memory
CB	Class-Based, tedy založený na třídách
CBWFQ	Class-Based Weighted Fair Queuing, třídní vážené fronty
CE	Customer Edge, hraniční směrovač sítě zákazníka
CFI	Canonical Format Indicator
CoS	Class of Service
CPU	Central Processing Unit – procesor
CQ	Custom Queuing – vlastní fronty
CRC	Cyclic Redundancy Check
cRTP	Enhanced Compressed RTP [10]
CS	Class Selector – selektor třídy
DFC	Distributed Forwarding Card
DHCP	Dynamic Host Configuration Protocol
DoS	Denial of Service, resp. odmítnutí služby
DRAM	Dynamic Random Access Memory
DS	Differentiated Services, DiffServ – diferencované služby
DSCP	Differentiated Services Code Point
ECN	Explicit Congestion Notification – explicitní oznamování zahlcení [21]
EF	Expedited Forwarding – urychlené doručování [21]
FCS	Frame Check Sequence – kontrolní součet v rámci protokolu ISL
FIFO	First In, First Out – fronty typu „první dovnitř, první ven“
FPGA	Field Programmable Gate Array
FRED	Fair Random Early Detection – spravedlivá náhodná včasná detekce
FTP	File Transfer Protocol – protokol pro přenos souborů, pracuje na TCP portu 20 a 21
FTP	File Transfer Protocol
GPP	General-Purpose Processor – procesory pro obecné použití
HDD	Hard Disk Drive
HP	Hewlett-Packard
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
HTTPS	HyperText Transfer Protocol Secure
ICMP	Internet Control Message Protocol
IEEE	Institute of Electrical and Electronics Engineers
IP	Internet Protocol
IPP	IP Precedence (IP priorita)
IPTV	televize po IP (Internet Protocol Television)
IPv4	Internet Protocol verze 4
IPv6	Internet Protocol verze 6
ISL	Inter-Switch Link
ISR	Integrated Services Router
ITU-T	International Telecommunication Union – Telecommunication Standardization Sector
Jitter	Proměnlivé zpoždění

Zkratka	Popis
LAN	Local Area Network, lokální síť
LLQ	Low-Latency Queuing – fronty s nízkým zpožděním
LQD	Longest Queue Drop, zahazování z nejdelší fronty
MAC	Media Access Control
MBS	Maximum Burst Size – maximální velikost shluku
mcc	počet spojení za sekundu (Maximum Concurrent Connections)
MDRR	Modified Deficit Round-Robin, modifikovaný deficitní okruh
MIME	Multipurpose Internet Mail Extensions – rozšiřující část záhlaví e-mailu
MPC	Modular Port Concentrator
MPD	Mark Probability Denominator – jmenovatel pravděpodobnosti označení
MPLS	Multi-Protocol Label Switching
mse	Mean Squared Error – střední kvadratická odchylka
MTU	Maximum Transmission Unit
NBAR	Network Based Application Recognition
NPU	Network (Processor) Processing Unit – síťový procesor
P	Provider, směrovače transportní sítě.
pcap	packet capture
PCP	pole priorita (Priority Code Point) v rámci 801.Q
PDF	Portable Document Format
PE	Provider Edge, hraniční směrovač sítě.
PFC	Policy Feature Card
PHB	Per-Hop Behaviors – chování při přeskoku
PQ	Priority Queuing – prioritní fronty
QoS	Kvalita služeb (Quality of Service)
QV	Quantum Value, hodnota kvanta, kvantum
RAM	Random Access Memory
RBACL	Roles-Based ACL
RDP	Remote Desktop Protocol
RED	Random Early Detection, náhodná včasná detekce
RFC	Request for Comments
RM ISO/OSI	Referenční model ISO (International Organization for Standardization)/OSI (Open Systems Interconnection model) [4].
RPM	Revolutions Per Minute – otáček za sekundu
RTCP	RTP Control Protocol
RTP	Real-Time Transport Protocol
SDH	Synchronous Digital Hierarchy – synchronní digitální hierarchie
SDN	Software-Defined Networking – softwarově definované sítě
SDRAM	Synchronous Dynamic Random Access Memory
SGT	Security Group Tag
SMDRR	Striktní MDRR
SOHO	Small Office/Home Office – malé a domácí kanceláře
SOM	Self Organizing Map – samoorganizující se mapy, tzv. Kohenenovy mapy
SPAN	Switched Port Analyzer
SRAM	Static Random Access Memory
SRED	Stabilized Random Early Detection – stabilizovaný RED
SSH	Secure Shell
SYN	SYNchronize
TC field	Traffic Class Field – původní pole EXP (experimental) v rámci MPLS
TCP	Transmission Control Protocol [65]
ToS	Type of Service (typ služby)
UDP	User Datagram Protocol [115]

Zkratka	Popis
UML	User Mode Linux
UPM	Universidad Politécnica de Madrid
URL	Uniform Resource Locator – internetová adresa
VID	VLAN identifier – identifikátor VLAN
VLAN	Virtuální LAN – vzájemné propojení virtuálních LAN sítí
VNUML	Virtual Network User Mode Linux
VOD	Video na vyžádání (Video on Demand)
VoIP	Voice over IP – označení služeb přenosu telefonu po IP sítích. Obvykle se jedná o protokoly SIP, RTP nebo H.323 a SCCP (Skinny Client Control Protocol) – protokol VoIP společnosti Cisco.
WDC	Western Digital
WFQ	Weighted Fair Queuing – vážené fronty
WRED	Weighted Random Early Detection, vážená náhodná včasná detekce
WRR	Weighted Roung-Robin – vážený Round-Robin – cyklická obsluha
WTD	Weighted Tail Drop – vážené zahazování posledních

SEZNAM SYMBOLŮ

Symbol	Popis
a_i	množina skutečných výstupů při trénování neuronové sítě
B	bajt – značení podle IEEE 1541 [116]
b	bit – značení podle IEEE 1541 [116] a IEEE 260.1 [117]
$\boldsymbol{B_c}$	Committed Burst Size (dle [21] „potvrzená velikost shluku“), (b/s)
$\boldsymbol{B_e}$	Excess Burst Size (dle [21] „velikost nadměrného shluku“), (b)
C	konfigurační matice
CIR	Committed Information Rate – potvrzená přenosová rychlost (b/s)
c_{p_i}	počet vstupních rámců
c_N	počet možných kombinací propojení vstupů na výstup
c_{Nv}	počet možných variací propojení vstupů na výstup
c_T	počet tokenů přidaných do tokenového zásobníku
c_{pi}	počet vstupních paketů
c_{po}	počet výstupních rámců/paketů
$\boldsymbol{d}$	pravděpodobnost zahození paketu
$\boldsymbol{h_c}$	celková hloubka fronty
$\boldsymbol{h_{max}}$	horní prahová hodnota
$\boldsymbol{h_{min}}$	spodní prahová hodnota
$\boldsymbol{h_p}$	průměrná hloubka fronty (počet paketů)
$\boldsymbol{h_0}$	okamžitá hloubka fronty (počet paketů)
i	pořadí prvku pravděpodobnostního vektoru
l_a	absolutní zpoždění [s]
l_i	změřená hodnota zpoždění (index i odpovídá pořadí měření)
l_n	pevné zpoždění [s]
l_v	proměnné zpoždění [s]
mse	Mean Squared Error – střední kvadratická odchylka
N	počet vstupů/výstupů přepínače
n	počet prvků
P	vstupní tréninková matice
p	paket
PIR	Peak Information Rate, špičková přenosová rychlost, (b/s)
p_l	ztrátovost paketů (%)
P_l	ztrátovost rámců (%)
$\boldsymbol{p_z}$	pravděpodobnost označení/přeznačkování paketu
R	velikost rámce (b)
s	sekunda
$\vec{\boldsymbol{s}}$	pravděpodobností vektor
$\bar{\boldsymbol{s}}$	směrodatná odchylka aritmetického průměru
$\boldsymbol{s_i}$	prvek pravděpodobnostního vektoru $\vec{s}$
T	cílová tréninková matice
$\boldsymbol{T_c}$	označuje časový interval, po který je možné odesílat shluk B_c, [s]
t_i	množina cílových hodnot trénování neuronové sítě
t_{p_n}	čas příchodu n paketu (rozdíl času dvou po sobě jdoucích paketů)
Tr	transakce, resp. transakce za sekundu (Tr/s)
v_p	přenosová rychlost (b/s)
v_{pk}	paketová rychlost (p/s)
v_r	rámcová rychlost (frame/s)
$\boldsymbol{\Delta c_p}$	počet paketů zařazených do fronty od posledního paketu označeného k zahození

SEZNAM PŘÍLOH

A PRIORITNÍ TŘÍDY PROTOKOLŮ

A.1 HODNOTY COS – ETHERNETOVÝ RÁMEC, PODLE IEEE 802.1Q [24]

Priorita	Zkratka	Typ provozu
1	BK	Provoz na pozadí (Background)
0	BE	Výchozí priorita (Best Effort)
2	EE	Zvýšená priorita (Excellent Effort)
3	CA	Kritické aplikace (Critical Applications)
4	VI	Video s latencí a jitterem < 100 ms
5	VO	Hlas s latencí a jitterem < 10 ms
6	IC	Mezisíťové řízení (Internetwork Control)
7	NC	Řízení sítě (Network Control)

A.2 HODNOTY COS – ETHERNETOVÝ RÁMEC, PROTOKOL CISCO ISL PODLE [25]

Priorita	Typ provozu
0	Normální provoz, výchozí priorita
1	Priorita 1
2	Priorita 2
3	Nejvyšší priorita

A.3 HODNOTY IPP – IPV4 HLAVIČKA PODLE RFC 791, STR. 12 [27]

Priorita	Typ provozu
0	Normální provoz, výchozí priorita (Routine)
1	Prioritní (Priority)
2	Okamžitý provoz (Immediate)
3	Bleskový provoz (Flash)
4	Bleskový prioritní (Flash Override)
5	Kritický (Critical)
6	Mezisíťové řízení (Internetwork Control)
7	Řízení sítě (Network Control)

A.4 HODNOTY DSCP – AF PODLE RFC 2597 [37]

Tabulka udává hodnoty pole DSCP. Tučně je zvýrazněna třída fronty (resp. priorita), následující 2 b označují pravděpodobnost zahození, poslední bit je vždy 0 a při vyjádření ve tvaru AFxx se neuvažuje [118].

Třída front (priorita)	Pravděpodobnost zahození		
	Nízká (Low Drop Precedence)	Střední (Medium Drop Precedence)	Vysoká (High Drop Precedence)
1	**001**010 (AF11)	**001**100 (AF12)	**001**110 (AF13)
2	**010**010 (AF21)	**010**100 (AF22)	**010**110 (AF23)
3	**011**010 (AF31)	**011**100 (AF32)	**011**110 (AF33)
4	**100**010 (AF41)	**100**100 (AF42)	**100**110 (AF43)

B Měření aktivních prvků

B.1 Měření ztrátovosti

Velikost rámce (B)	Přenosová rychlost (%)	Ztrátovost (%)				
		VNUML	Catalyst 2960	Micronet SP608K	HP Pro-Curve	3Com
64	100	0,000		0,000	0,088	
	90	0,000		0,000	0,000	
	80				0,000	
128	100	0,000	0,114	0,000	0,095	94,966
	90	0,000	0,000	0,000	0,000	95,510
	80		0,000		0,000	94,774
	70					84,112
	60					73,401
	50					63,620
	40					0,000
	30					0,000
256	100	0,000	0,114	0,000	0,102	94,834
	90	0,000	0,000	0,000	0,000	96,267
	80		0,000		0,000	89,073
	70					81,612
	60					71,793
	50					58,725
	40					34,544
	30					0,000
	20					0,000
512	100	0,000	0,104	0,000	0,071	94,918
	90	0,000	0,000	0,000	0,000	93,926
	80		0,000		0,000	87,546
	70					80,541
	60					68,500
	50					52,553
	40					0,000
	30					0,000
1024	100	0,000	0,116	0,000	0,081	95,270
	90	0,000	0,000	0,000	0,000	93,761
	80		0,000		0,000	85,417
	70					76,811
	60					63,850
	50					40,519
	40					0,000
	30					0,000
1280	100	0,000	0,116	0,000	0,105	95,215
	90	0,000	0,000	0,000	0,000	93,411
	80		0,000		0,000	84,362
	70					74,430
	60					60,315
	50					36,743
	40					0,000
	30					0,000
1518	100	0,000	0,116	0,000	0,114	95,074

Velikost rámce (B)	Přenosová rychlost (%)	Ztrátovost (%)				
		VNUML	Catalyst 2960	Micronet SP608K	HP Pro-Curve	3Com
	90	0,000	0,000	0,000	0,000	92,891
	80		0,000		0,000	83,026
	70					72,826
	60					56,839
	50					29,014
	40					0,000
	30					0,000

C Neuronové sítě

C.1 Terminologie

Prioritní vektor: jeho velikost je určena počtem vstupních front [90], [102].

Konfigurační vektor (konfigurační vzor) určuje, které prvky spínací matice jsou sepnuty [90], [102].

Neuron je charakterizován vstupním a výstupním vektorem [90], [102].

SOM (Self Organizing Map) – samoorganizující se, tzv. Kohenenovy, mapy [90], [102].

V neuronové síti lze rozlišovat 3 základní typy neuronů [90], [102]:

- § vstupní,
- § pracovní, často nazývaní skryté nebo mezilehlé či vnitřní. Anglicky je tato vrstva neuronů nazývána hidden layer,
- § výstupní (output layer).

Stav neuronové sítě určují stavy všech neuronů. Šíření a zpracování informace je umožněno změnou stavu neuronu [90], [102].

Konfigurace neuronové sítě je dána sympatickými váhami všech spojů [90], [102].

Neuronová síť má **tři režimy práce**. Tyto režimy se také nazývají „dynamika neuronové sítě“ [90], [102]:

- § organizační – mění se topologie sítě,
- § aktivní – mění se stav,
- § adaptivní – mění se konfigurace.

Neuronová síť může být z topologického hlediska [90], [102]:

- § **cyklická** (reduktivní),
- § **acyklická** (dopředná). Dopřednou síť lze vždy rozdělit do vrstev. Spoje mezi neurony vedou vždy jen z vrstev nižších do vyšších. Spoje ale mohou jednu i více vrstev přeskočit. Ve vícevrstvové síti jsou neurony jedné vrstvy spojeny se všemi neurony vrstvy bezprostředně následující.

Vstupní prostor sítě – všechny možné vstupy sítě [90], [102].

Stavový prostor sítě – všechny možné stavy sítě [90], [102].

Synchronní model sítě – neurony mění svůj stav nezávisle [90], [102].

Asynchronní model – aktualizace, tedy změna, stavu neuronů je řízena centrálně [90], [102].

Bias – váhová hodnota přiřazená vstupu neuronu, jehož aktivace (vstup) je 1 [90], [102].

Vícevrstvá síť – neuronová síť s více vrstvami, např. Madaline, feedforward backpropagation atd [90], [102].

C.2 Problémy nástroje nntool

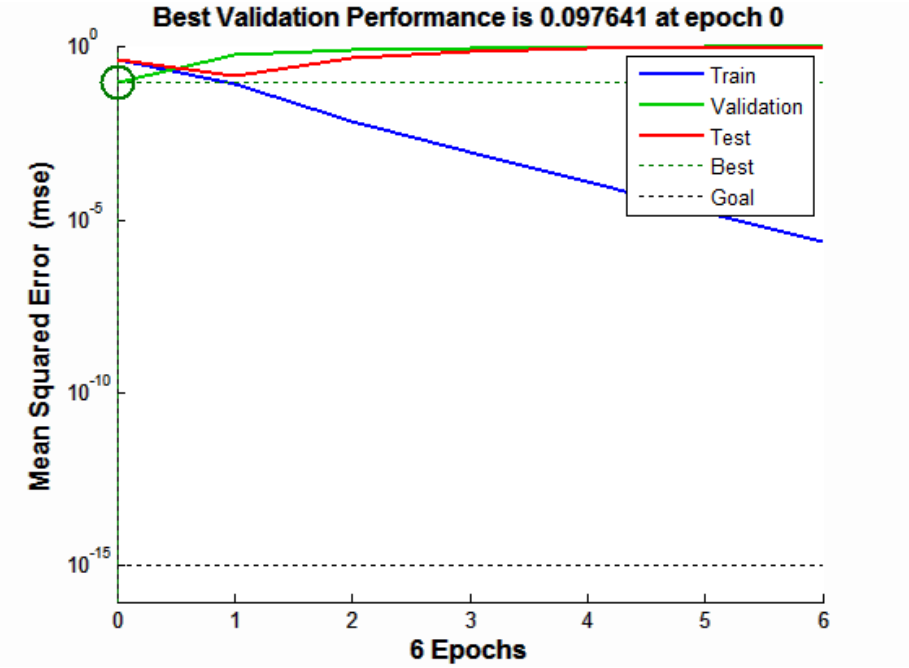
Vzhledem k problémům s neočekávaně nízkou přesností výsledků dosažených při modelování neuronových sítí s použitím nástroje nntool, byl záhy zahájen výzkum příčin popsaných problémů. Nástrojem nntool byla vytvořena neuronová síť typu feedforward backpropagation

určená k řešení XOR, popsaná v kap. 3.3.3, obr. 3.7. Celý postup byl odvozen od postupu vytvoření a trénování sítě podle literatury [89], viz rovněž kap. 1.6.3.

Trénování vytvořené sítě metodou trainlm bylo provedeno s de facto výchozím nastavením nástrojem nntool. Síť byla trénována maximálně 1 000 epochami, cílová chybovost, resp. výkon byl nastaven na $1 \cdot 10^{-15}$, minimální gradient (`min_grad`) byl nastaven na $1 \cdot 10^{-10}$. Výsledky, které byly dosaženy, neodpovídaly očekávaným. Vstupní rozsah dat byl odvozen od vstupní matice p, viz rovnice (3.8) a měl tvar matice (6.1).

$$input\ range = \begin{pmatrix} 0 & 1 \\ 0 & 1 \end{pmatrix} \tag{6.1}$$

Trénování sítě při opakovaných pokusech s náhodně nastavenými vahami a biasem pro jednotlivé neurony končilo po přibližně 6 epochách tréninku. Příčinou byla malá změna gradientu, tedy již zanedbatelné zlepšování vývoje sítě. Nastavení větší tolerance (zmenšení gradientu) vedlo podle očekávání pouze k prodloužení trénování, ne však k lepším výsledkům. Výsledky trénování jsou zřejmé z grafu na obr. C.1.



Obr. C.1: Trénování XOR pomocí nntool – vývoj mse v průběhu trénování sítě

Z grafu (obr. C.1) je patrné, že dosažené výsledky trénování jsou nevyhovující. Nejlepší dosažený výkon (resp. chybovost mse) byla dosažena 0,0097641. Problém je také vidět z výsledků ověřovací množiny (6.2), která je výsledkem zpracování vstupní matice (3.8) vytvořenou neuronovou sítí. Očekávaným výsledkem byla matice (3.9).

$$output\ data = (0{,}66251 \quad 0{,}98097 \quad 0{,}68752 \quad 0{,}95971) \tag{6.2}$$

Nedostatečnost dosažených výsledků je také patrná z chybové matice, resp. matice odchylek od očekávaného stavu (6.3). Chyba je mnohdy téměř 100%.

$$error\ data = (-0{,}66251 \quad 0{,}019034 \quad 0{,}31248 \quad -0{,}95971) \tag{6.3}$$

Vzhledem k tomu, že dosažené výsledky byly zcela neuspokojivé, byla zkoumána jejich příčina. Postupně se ukázalo, že literatura [89] je zastaralá a nereflektuje změny, ke kterým došlo

v posledních verzích Matlabu (2010b). V různých verzích Matlabu jsou různým způsobem nastaveny výchozí parametry trénování vytvořené sítě. Dále se ukázalo, že nástroj nntool je taktéž nevyhovující, protože nastavení některých parametrů trénování sítě vůbec neumožňuje.

Zmíněné parametry učícího algoritmu v nástroji nntool nelze nastavit a výchozí hodnoty parametrů jsou odlišné od žádaných. Z toho plyne, že nástrojem nntool nelze v Matlabu 2010b standardně natrénovat síť s ekvivalentní funkčností k logické funkci XOR.

Určitým možným řešením problému nastavitelnosti nástroje nntool je zvětšení tréninkové množiny, např. její několikanásobnou duplikací. Pak výchozí nastavení nntool, které používá pro trénování 70 % ze vstupní matice, pro ověření dat 15 % a pro testování zbývajících 15 % nevadí a lze úspěšně natrénovat síť s požadovanými vlastnostmi.

Kvůli popsaným problémům bylo použití nástroje nntool opuštěno a veškeré nové neuronové sítě byly vytvářeny výhradně pomocí skriptů, které umožňují kontrolovat všechny potřebné parametry vytvářené neuronové sítě.

C.3 Algoritmus učení Levenberg – Marquardt

Parametry, které je možné nastavit při použití metody TrainLM jsou (v závorkách za jednotlivými parametry je uvedena jejich výchozí hodnota) [90]:

`Epochs` (100) – maximální počet tréninkových epoch.

`Goal` (0) – výkonový cíl. Nejlepší výkon je dosažen, pokud je roven 0. Musí být kladné číslo. Výkon se měří jako [90]:

- `mae` – průměrná absolutní chyba (mean absolute error),
- `mse` – střední kvadratická chyba (mean squared normalized error), Mse je nejčastěji používaná metoda měření výkonu.
- `sse` – součet střední kvadratické chyby (sum squared error).

`Max_fail` (5) – maximum chyb při validaci.

`Mem_reduc` (1) – poměr mezi spotřebovanou pamětí a rychlostí. Pokud je mem_reduc nastaven na 1, je metoda rychlá, ale potřebuje hodně paměti. Pokud je mem_reduc nastaveno na 2, sníží spotřebu paměti, ale algoritmus je pomalejší…

`Min_grad` (1e-10) – minimální „performance gradient“, tedy nejmenší gradient výkonu než je trénování zastaveno. Pokud se výkon neuronové sítě zlepšuje pomalu, je významné zlepšení nepravděpodobné a trénování se zbytečně prodlužuje.

`Mu` (0,001) – počáteční mu. Mu je počáteční hodnota pro μ. Algoritmus Levenberg – Marquardt používá Hessianovu matici, která tvoří jádro tohoto učícího algoritmu. Algoritmus umožňuje rychle dospět k žádanému výkonu sítě [90].

`Mu_dec` (0,1) – faktor snižování mu (krok o který se bude snižovat mu, resp. konstanta, kterou se bude mu násobit při jeho snižování).

`Mu_inc` (10) – faktor zvyšování mu (krok o který se mu bude zvyšovat, resp. konstanta, kterou se bude mu násobit při jeho zvyšování).

`Mu_max` (1e10) – maximální mu.

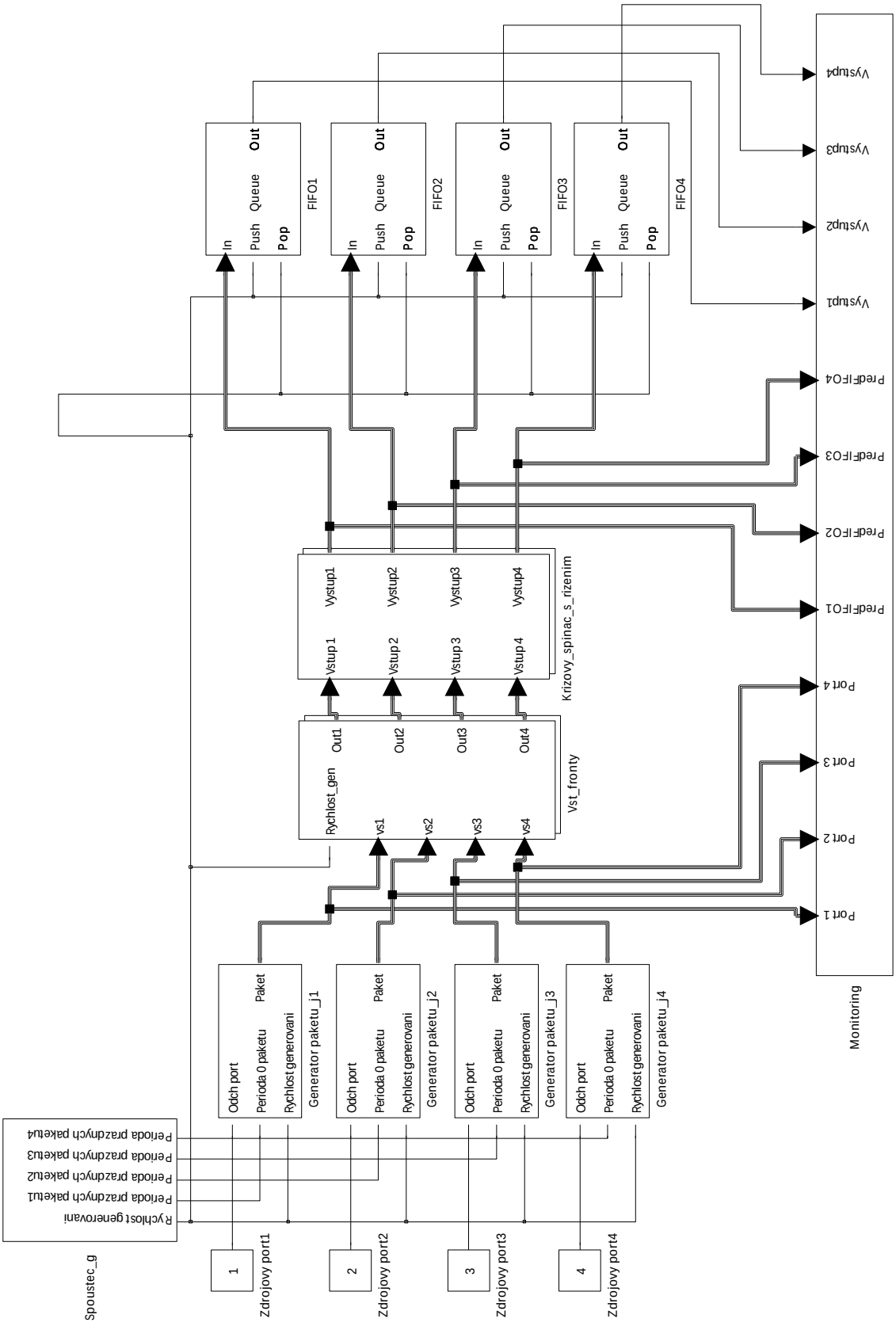
D VNUML

D.1 KONFIGURAČNÍ XML SKRIPT

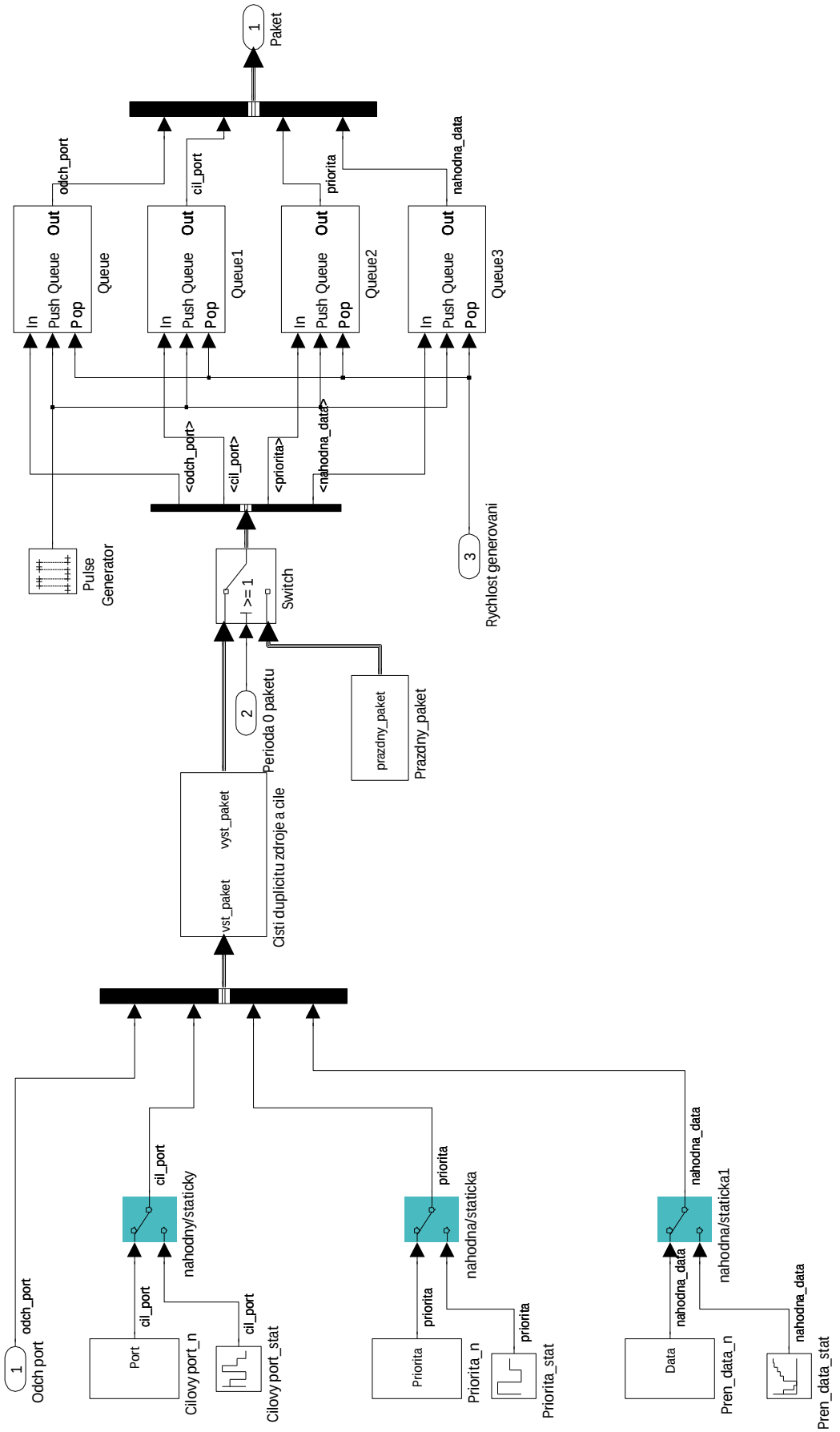
```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE vnuml SYSTEM "/usr/share/xml/vnuml/vnuml.dtd">
<!--
Topologie hvezdy - 3 uzly pripojene k centralnimu switchi bez vnejsiho
propojeni.
-->
<vnuml>
  <global>
    <version>1.8</version>
    <simulation_name>hvezda</simulation_name>
    <automac/>
    <vm_mgmt type="none" />
    <vm_defaults exec_mode="mconsole">
       <filesystem
type="cow">/usr/share/vnuml/filesystems/root_fs_tutorial</filesystem>
       <kernel>/usr/share/vnuml/kernels/linux</kernel>
       <console id="0">xterm</console>
       <!--xterm>gnome-terminal,-t,-x</xterm-->
    </vm_defaults>
  </global>
  <net name="Net0" mode="uml_switch"
uml_switch_binary="/usr/local/vnuml_mod/uml_switch4capture" type="lan"/>
  <vm name="uml1">
    <if id="1" net="Net0">
      <ipv4>192.0.0.1</ipv4>
    </if>
    <route type="ipv4" gw="192.0.0.3">default</route>
    <exec seq="start" type="verbatim">nohup /usr/bin/hello &lt;/dev/null
&gt;/dev/null 2&gt;&amp;1 &amp; </exec>
    <exec seq="stop" type="verbatim">killall hello</exec>
  </vm>
  <vm name="uml2">
    <if id="1" net="Net0">
      <ipv4>192.0.0.2</ipv4>
    </if>
    <route type="ipv4" gw="192.0.0.3">default</route>
  </vm>
  <vm name="uml3">
    <if id="1" net="Net0">
      <ipv4>192.0.0.3</ipv4>
    </if>
    <route type="ipv4" gw="192.0.1.2">192.0.2.0/24</route>
    <forwarding type="ip" />
  </vm>
</vnuml>
```

E KOMPLEXNÍ SIMULAČNÍ MODEL

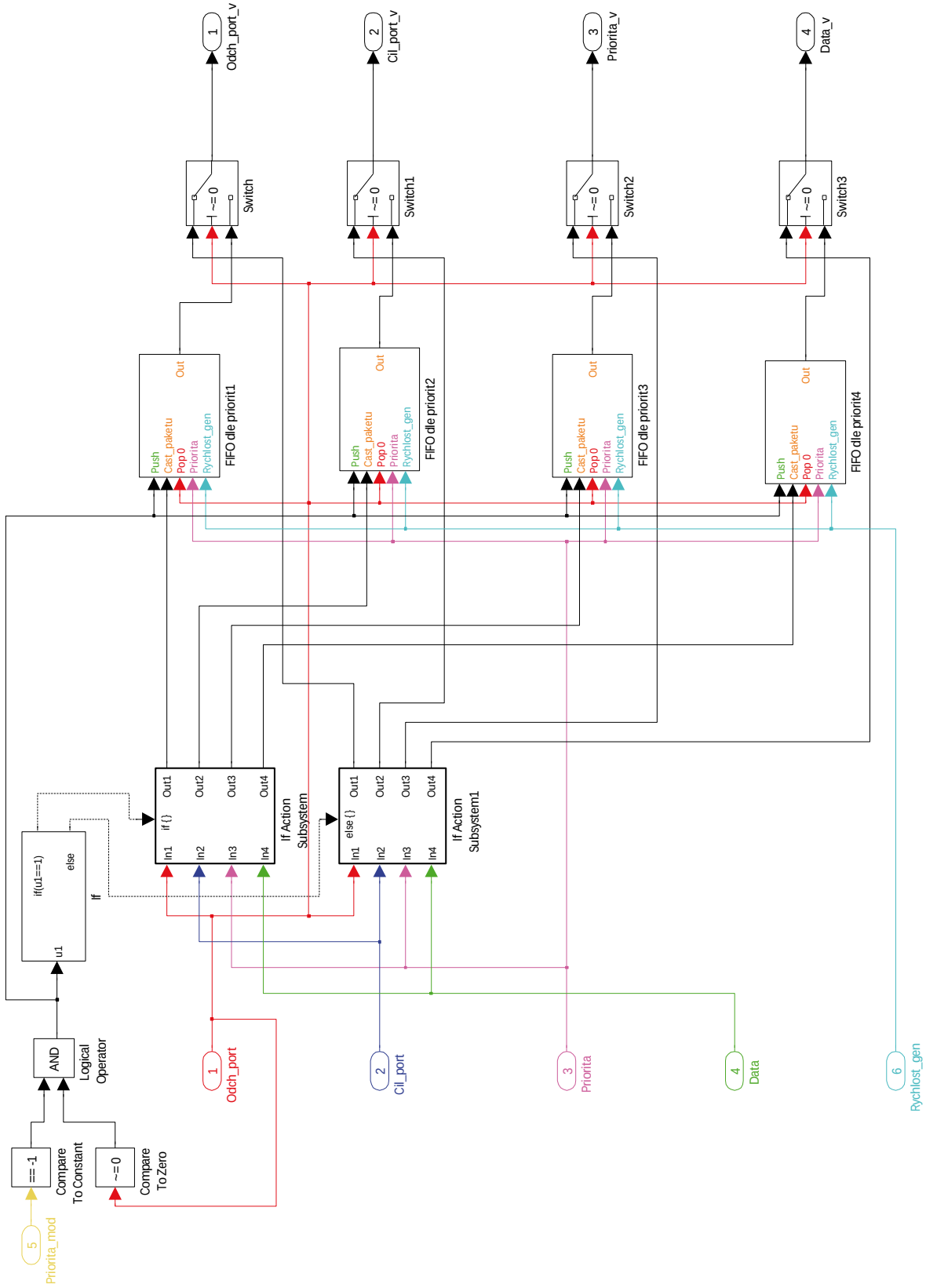
E.1 PRVNÍ VRSTVA KOMPLEXNÍHO MODELU



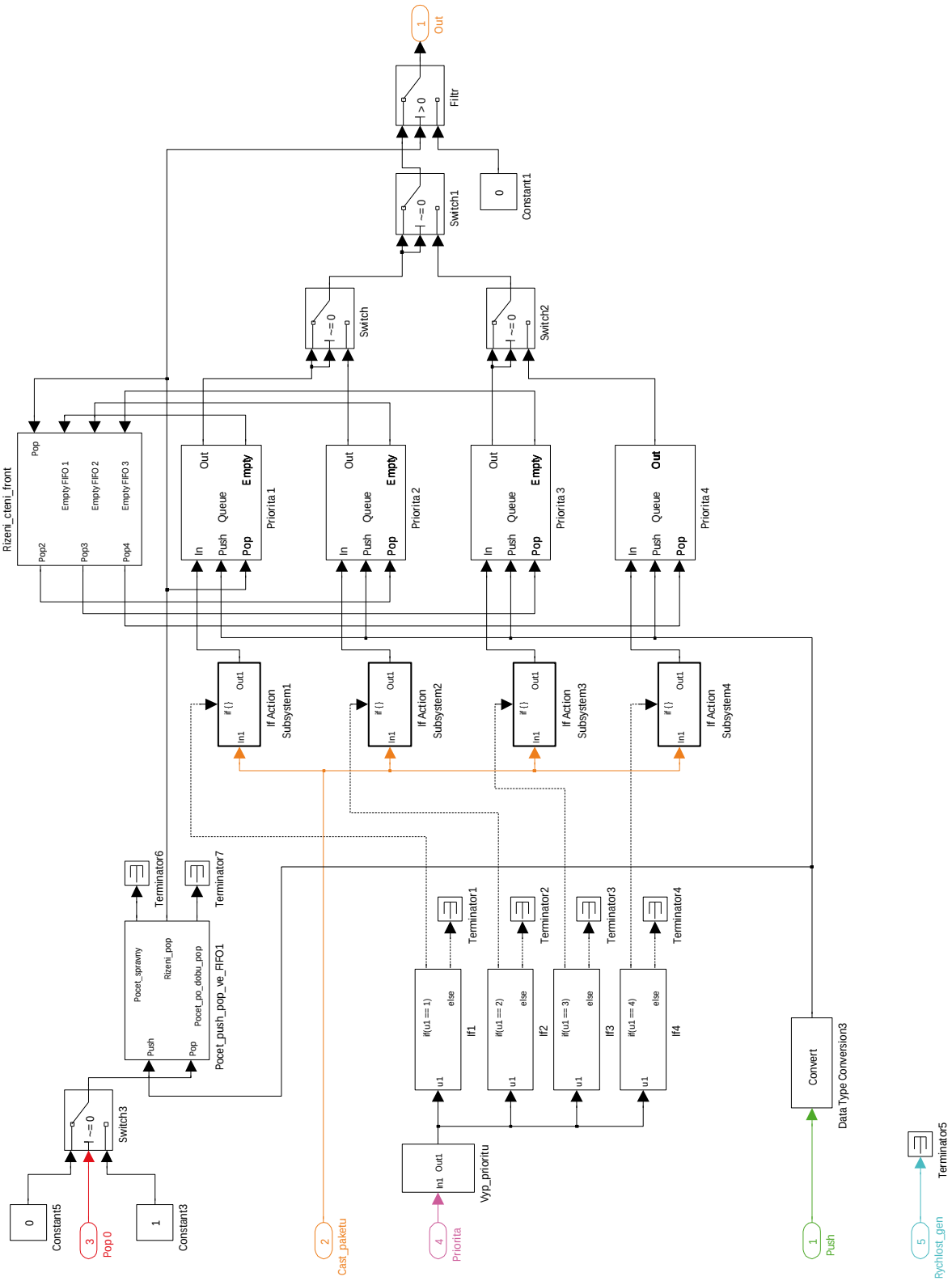
E.2 BLOKOVÉ SCHÉMA GENERÁTORU PAKETŮ



E.3 Blokové schéma správy vstupních front



E.4 Blokové schéma správy front algoritmem SMDRR



SEZNAM VLASTNÍCH PRACÍ

[1] POLÍVKA, Michal; ŠKORPIL, Vladislav. Přístupové a transportní sítě – laboratorní cvičení. Brno: Vysoké učení technické v Brně, 2013. s. 1-80. ISBN: 978-80-214-4726- 4.

[2] ŠKORPIL, Vladislav; POLÍVKA, Michal. Parallel WAN Switch Based on Neural Network. In Recent Researches in Communi-cations and Computers. WSEAS Press, 2012. s. 68-72. ISBN: 978-1-61804-109- 8.

[3] POLÍVKA, Michal; ŠKORPIL, Vladislav. Modeling of Network Switch Controlled by Neural Network. In 2012 35th Internatio-nal Conference on Telecommunications and Signal Processing (TSP) (id 20316). 2012. s. 165-168. ISBN: 978-1-4673-1116- 8.

[4] POLÍVKA, Michal; ŠKORPIL, Vladislav. Modeling Logical Function Antivalence Using Neural Network in MATLAB. In 2011. ISBN: 978-80-214-4283- 2.

[5] POLÍVKA, Michal; PELKA, Tomáš. Computer network switch testing and modelization. Elektrorevue – Internetový časopis (http://www.elektrorevue.cz/), 2011, roč. 2011, č. 2, s. 32-38. ISSN: 1213- 1539.

[6] POLÍVKA, Michal; PERROTON, Laurent; PELKA, Tomáš. VNUML – application in computer network, switch modeling. In 6th Inter-national Conference on Teleinformatics. 2011. s. 136-141. ISBN: 978-80-214-4231- 3.

[7] POLÍVKA, Michal; PELKA, Tomáš. Unified Communications – Possible Risks and Their Solutions. In Telecommunications and signal processing. 2010. s. 1-4. ISBN: 978-963-88981-0-4.

[8] POLÍVKA, Michal; PELKA, Tomáš. Simulátor rozsáhlých sítí – SimPP. Elektrorevue – Internetový časopis (http://www.elektrorevue.cz/), 2010, roč. 2010, č. 9, s. 1-5. ISSN: 1213- 1539.

[9] PELKA, Tomáš; POLÍVKA, Michal; HAJNÝ, Jan. SimPP – Wide Area Network Simulator. In The 33rd International Conference on Telecommunication and Signal Processing, TSP 2010. 2010. s. 1-4. ISBN: 978-963-88981-0- 4.

[10] POLÍVKA, Michal; PELKA, Tomáš; KACÁLEK, Jan. Moderní telekomunikační služby ve výuce. In International WORKSHOP RTT 2009 Research in Telecommunication Technology. 2009. s. 1-3. ISBN: 978-80-01-04411- 7.

[11] PELKA, Tomáš; POLÍVKA, Michal. Clustering a jeho využití. In International WORKSHOP RTT 2009 Research in Telecom-munication Technology. 2009. s. 1-4. ISBN: 978-80-01-04411-7.

[12] PELKA, Tomáš; POLÍVKA, Michal. Comparison of Python virtual machines. Linux+, 2008, roč. 2008, č. 11, s. 1-8. ISSN: 1733- 4209.

[13] POLÍVKA, Michal. Možnosti moderního síťového zabezpečení zastaralé výpočetní techniky. Elektrorevue – Internetový časopis (http://www.elektrorevue.cz/), 2008, roč. 2008, č. 27, s. 1-7. ISSN: 1213- 1539.

[14] POLÍVKA, Michal; PELKA, Tomáš; KYSELÁK, Martin. Using terminal server in education. In 9-th International Conference – Re-search in Telecommunication Technology RTT - 2008. 1. 2008. s. 1-7. ISBN: 978-80-227-2939- 0.

[15] PELKA, Tomáš; POLÍVKA, Michal. AAA protokoly. Hakin9, 2008, roč. 2008, č. 3, s. 48-52. ISSN: 1214- 7710.

[16] ŠKORPIL, Vladislav; MOLNÁR, Karol; NOVOTNÝ, Vít; JEŘÁBEK, Jan; HANUS, Stanislav; POLÍVKA, Michal; ČÍKA, Petr; ZEMAN, Václav.; ČÍŽ, Radim. Encyklopedie Teleinformatiky. Encyklopedie Teleinformatiky. VUT. Brno: VUT Brno, 2008. s. 1-751.

Životopis

Vzdělání

2007–dosud student Vysokého učení technického v Brně (http://www.vutbr.cz/), **Fakulta elektrotechniky a komunikačních technologií** – doktorský studijní program (elektrotechnika a komunikační technologie – obor teleinformatika).

2009 pětiměsíční stáž na Université Paris-Est, LIGM, Equipe A3SI, Computer Science Dept. **ESIEE Paris** (L'École d'ingénieurs des sciencés et technologies de l'information et de la communication), Cité Descartes, BP 99, 93162 Noisy-le-Grand Cedex France, http://www.esiee.fr/ – projekt v rámci doktorského studia.

2009–2010 doplňující **pedagogické studium** pro zaměstnance VUT.

2001–2007 Vysoké učení technické v Brně (http://www.vutbr.cz/), **Fakulta elektrotechniky a komunikačních technologií** – magisterský studijní program (elektrotechnika a informatika – obor elektronika a sdělovací technika).

Účast na projektech

2011 člen projektového týmu „**Výzkum elektronických komunikačních systémů**“, FEKT-S-11-15, Vysoké učení technické v Brně – Vnitřní projekty VUT – plně financující (01. 01. 2012 – 31. 12. 2014).

2010 spoluřešitel projektu „**Simulátor transportních sítí**“, 3046/F1a, Ministerstvo školství, mládeže a tělovýchovy ČR – Fond rozvoje vysokých škol (FRVŠ) – plně financující (01. 01. 2010 – 31. 12. 2010).

2010 člen projektového týmu „**Výzkum komunikačních systémů a sítí**“, FEKT-S-10-16, Vysoké učení technické v Brně - Vnitřní projekty VUT – plně financující (01. 01. 2010 – 31. 12. 2010)

2009 hlavní řešitel projektu „**Inovace stávajících a příprava nových laboratorních úloh předmětu Služby telekomunikačních sítí**“, 1033/G1, Ministerstvo školství, mládeže a tělovýchovy ČR – Fond rozvoje vysokých škol (FRVŠ) – plně financující.