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

SIMULACE SÍŤOVÉHO PRVKU V PROSTŘEDÍ MATLAB.

SIMULATION OF NETWORK ELEMENT IN MATLAB ENVIRONMENT

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

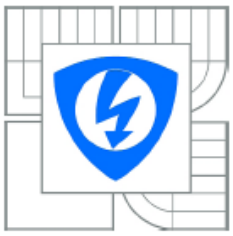
AUTOR PRÁCE Bc. PETER KUCHÁR

AUTHOR

VEDOUCÍ PRÁCE Ing. PETR VYCHODIL

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ
TECHNICKÉ V BRNĚ

Fakulta elektrotechniky
a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Peter Kuchár	***ID:***	72944
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Simulace síťového prvku v prostředí Matlab

POKYNY PRO VYPRACOVÁNÍ:

Prostudujte možnosti využití neuronových sítí pro řízení moderního síťového prvku s podporou kvality služeb. Zvolte vhodný síťový prvek a tento prvek realizujte v prostředí MATLAB-SIMULINK. Proveďte měření zajímavých charakteristik tohoto nasimulovaného prvku. Naměřené hodnoty diskutujte s veličinami u běžně dostupných síťových prvků.

DOPORUČENÁ LITERATURA:

[1] ZEDNÍČEK, P. Síťový prvek s pokročilým řízením. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2010. 70 s.
[2] DEMUTH, H.; BEALE, M. Neural Network Toolbox for Use with MATLAB. Natick (USA) : MathWorks, Inc., 1994
[3] DRÁBEK, O.; TAUFER, I; SEIDL, P. Umělé neuronové sítě - teorie a aplikace (5). CHEMagazín 5 (XVI), 2006, s. 6-8. ISSN 1210-7409
[4] DRÁBEK, O.; TAUFER, I; SEIDL, P. Umělé neuronové sítě - teorie a aplikace (6). CHEMagazín 6 (XVI), 2006, s.31-33. ISSN 1210-7409

Termín zadání:	7.2.2011	***Termín odevzdání:***	26.5.2011

Vedoucí práce: Ing. Petr Vychodil

prof. Ing. Kamil Vrba, CSc.
Předseda oborové rady

UPOZORNĚNÍ:

Abstrakt

Diplomová práca je venovaná problematike sieťových prvkov. V úvode dokumentu sú popísané technické aspekty ako je vnútorná štruktúra sieťových prvkov a riadiace mechanizmy slúžiace k rozlíšeniu služieb a podpore ich kvalitatívnych požiadaviek. V prvej kapitole sú uvedené riadiace mechanizmy od tých najjednoduchších, ako je FIFO, až po sofistikované, akým je CBWFQ. Opomenuté nie sú ani druhy aktívneho manažmentu front, ktoré sú rozobraté v časti Riadiaci člen. Veľkú úlohu pri rozlíšení služieb hrajú procesy značenia a klasifikácie paketov. Druhá kapitola je venovaná najperspektívnejšiemu štandardu DiffServ. Obsahuje časti venované architektúre, značkovaniu paketov, kódovému slovu DSCP a záver kapitoly je venovaný spôsobom správania počas preskoku a to prednostné odoslanie EF a technike zaručeného odoslania AF. Tretia kapitola uvádza prehľad najbežnejších neurónových sietí, ich vlastnosti a posudzuje ich vhodnosť nasadenia v sieťovom prvku. Samotný návrh smerovača a jeho konštrukcia v programovom prostredí Matlab/Simulink je predmetom štvrtej kapitoly. Okrem použitých blokov z knižníc Simulinku je uvedené ich nastavenie a funkcia v zapojení. Následne sú zhodnotené výsledky a vyvodené závery.

Kľúčové slová: Sieťový prvok, Simulink, spojovacie pole, aktívny manažment front, DiffServ, WRED

Abstract

Master's thesis is dedicated to the issue of network element. In the first part are described technological aspects as the internal structure of network elements and control mechanisms that provide differentiaton of services and support their quality requirements. In the first chapter are listed control mechanisms from most simple like FIFO to the more sophisticated like CBWFQ. Active queue managements are not missing and they are described in the section Riadiaci člen. Significant role in the differentiation of services have processes marking and pakets classification. The second chapter is devoted most promising standard DiffServ. Contains section devoted to architecture, paket marking, code point DSCP and the final part is devoted to the types of per-hop behavior and it is expedited forwarding EF and technique assured forwarding AF. The third chapter gives an overview common models of neural networks, their properties and assess their suitability for deployment in network elements. The routher design itself and its structure in programming environment Matlab/Simulink is the subject of the fourth chapter. Except used bloks from Simulink library is described their setting and function in the wiring. Consequently results are reviewied and conclusions drawn.

Keywords: Network element, Simulink, switching array, active queue management, DiffServ, WRED

KUCHÁR, P. *Simulace síťového prvku v prostředí Matlab.* Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, 2011. 64s. Vedoucí diplomové práce Ing. Petr Vychodil.

Prohlášení

Prohlašuji, že svou diplomovou práci na téma SIMULACE SÍŤOVÉHO PRVKU V PROSTŘEDÍ MATLAB jsem vypracoval samostatně pod vedením vedoucího diplomové práce 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 diplomové práce dále prohlašuji, že v souvislosti s vytvořením tohoto projektu jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č.140/1961 Sb.

V Brně dne ....

podpis autora

Obsah

ÚVOD

Diplomová práca je venovaná problematike riadenia a funkčnosti sieťových prvkov. Sieťové prvky v dnešnej dobe tvoria jeden z kľúčových elementov všetkých komunikačných sietí. Vlastnosti sieťového prvku majú zásadný vplyv na celú sieť a jej prenosové vlastnosti. S rastúcim množstvom aplikácií vyžadujúcich prenos dát v reálnom čase sa zvýšil tlak na prenosové siete. Po zvýšení prenosových rýchlostí liniek prišiel rad na úzke miesta v podobe prepojovacích uzlov.

Po dosiahnutí optimálnej hardvérovej štruktúry a technologického stropu vyhotovenia, zostal priestor na zlepšenie vlastností v riadiacom mechanizme. Riadenie sieťových prvkov, správa front a zaobchádzanie s dátami rôznych prioritných tried je hlavnou náplňou mnohých výskumov. Posledné roky sa začalo uvažovať o využití neurónových sietí v riadení sieťových prvkov za účelom dosiahnutia lepších výsledkov, hlavne v oblasti smerovania. Fyzická implementácia do sieťového prvku pre skúšobné potreby je príliš nákladná a nepraktická, preto sa začala používať rada simulačných programov. Práca sa venuje práve tejto oblasti a rozoberá vnútornú štruktúru sieťových prvkov, ako aj metódy správy front a zaobchádzanie s dátami rôznych priorít. V tejto oblasti má najväčšiu perspektívu technológia DiffServ, ktorej je venovaná druhá kapitola a jej prvky sú použité aj v simulovanom prvku. Do neurónových sietí sú vkladané veľké nádeje a preto je v práci zahrnutý úvod do problematiky a popis jednotlivých druhov sietí, ich vlastností a vhodnosť ich použitia v sieťovom prvku. K overeniu teoretických predpokladov a záverov je použité programové prostredie Matlab so svojou nadstavbou Simulink. V programovom prostredí bude z funkčných blokov knižníc Simulinku zostavený sieťový prvok typu smerovač, v ktorom budú použité vybraté technológie z predchádzajúceho rozboru. Popisu konštrukcie, nastaveniu blokov a výsledkom simulovaného sieťového prvku sa venuje štvrtá a piata kapitola.

1. Sieťový prvok a jeho vnútorná štruktúra

V tejto kapitole bude všeobecne popísaný sieťový prvok, jeho druhy a funkčné bloky vnútornej štruktúry prvku. Venované to bude výhradne aktívnym sieťovým prvkom, kde nás najviac budú zaujímať sieťové prvky pracujúce na tretej a štvrtej vrstve modelu ISO/OSI, teda sa jedná o smerovač (Router) s podporou pre zabezpečenie kvality služieb (QoS). Každá technológia zaisťujúca kvalitu služieb musí vykonávať dve základné úlohy:

A triedenie dátových jednotiek podľa parametru provozu alebo služby
B zaistenie odlišného zaobchádzania s jednotlivými triedami, riadeným prideľovaním sieťových zdrojov.

Úloha A, triedenie dátových jednotiek sa bežne prevádza na rozhraní medzi užívateľom a sieťou alebo na rozhraní medzi sieťovými prvkami. Táto úloha sa delí na dve základné funkcie:

- *klasifikácia* paketu na základe identifikátoru provozu alebo typu triedy
- *značkovanie* paketu pridelením identifikátoru triedy, kam bol paket zaradený. Slúži na urýchlenie prevedenia nasledujúcich funkcií.

Úloha B, odlišné zaobchádzanie s paketmi bežne poskytujú aktívne sieťové prvky, predovšetkým smerovače v štyroch hlavných funkciách:

- *dohľad nad provozom* meria prichádzajúci provoz a prípadne vyradí alebo preznačí dátové jednotky vybočujúce z dohodnutých parametrov
- *aktívna správa fronty*, jej pomocou je dosiahnuté rozdielne zaobchádzanie s jednotlivými triedami
- *plánované odosielanie* rozhoduje, v ktorom okamžiku bude, ktorý paket vyslaný
- *tvarování provozu*, má za úlohu vyhladzovať zhlukový charakter prenosu čo zabezpečí lepšie využitie dostupnej kapacity linky [6][1].

Každý z prepojovacích uzlov sa dá rozložiť na tieto základné funkčné bloky:

- Vstupno – výstupné porty
- Vyrovnávacia pamäť (Buffer)
- Spojovacie pole
- Riadiaci člen

1.1 Vstupno – výstupné porty

Jedná sa o hardwarové vstupy slúžiace na pripojenie prenosových médií, ktoré sú zakončené príslušným konektorom podľa typu prenosového média a smerovača. Najčastejšie sa jedná v prípade ethernet smerovačov o konektor RJ 45 a v prípade optického smerovača o konektory MJ-RJ, LC a SC-type.

1.2 Vyrovnávacia pamäť

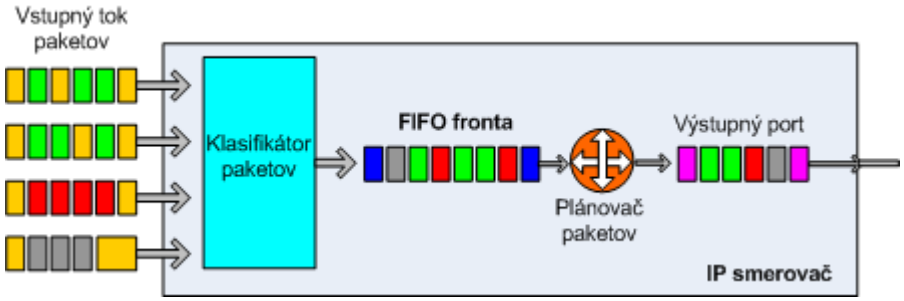
Do smerovača prichádzajú pakety na vstupné porty po vedeniach s rôznou rýchlosťou a kapacitou. Pakety sú smerované na odpovedajúce výstupné porty, resp. výstupný port, pričom v prípade obsadenia výstupného portu sa radia do fronty. Pre funkciu fronty môže byť vyrovnávacia pamäť umiestnená na vstupe, na výstupe alebo zdieľaná pre obidva rozhrania sieťového prvku. Najvhodnejšie varianty sú prepojovací prvok so vstupnou aj výstupnou vyrovnávacou pamäťou a riešenie virtuálnych výstupných front (VOQ). V prvom prípade sú pamäte umiestnené na oboch koncoch zariadenia a v druhom sú vo vstupnej pamäti pre každý port vytvorené virtuálne výstupné fronty odpovedajúce každá jednému výstupnému portu. V prípade zaistenia podpory kvality služieb je ešte možné rozdelenie na fronty, kde každá fronta odpovedá jednej triede kvality. Táto problematika rozdelenia ponúka viaceré možné riešenia[1][2]:

- Metóda obsluhy v poradí príchodu FIFO (*First-In-First-Out*)
- Metóda priameho radenia do front PQ (*Priority Queuing*)
- Metóda spravodlivého radenia do front FR (*Fair Queuing*)
- Metóda váhovej cyklickej obsluhy WRR (*Weighted Round Robin*)
- Metóda váhového spravodlivého radenia do front WFQ (*Weighted Fair Queuing*)
- Metóda WFQ založená na triedení prevádzky CB (*Class-Based WFQ*)

Metóda obsluhy v poradí príchodu FIFO

Metóda FIFO predstavuje predvolený mechanizmus radenia do front bez prítomnosti špecifického algoritmu plánovania paketov. Vo FIFO sa pakety radia do jedného radu v poradí v akom prišli a sú vysielané na výstupné porty v tom istom poradí, v akom sa zaradili do fronty (obr. 1.1). Hlavnou výhodou tejto metódy je jednoduchosť tohto mechanizmu. Všetko čo potrebuje táto metóda je pamäť v podobe jednoduchého zásobníka, ktorý umožňuje ukladanie prichádzajúcich paketov a ich vysielanie v rovnakom poradí v akom prišli. FIFO pristupuje ku všetkým paketom rovnako a preto sa najlepšie hodí pre „best effort“ siete.

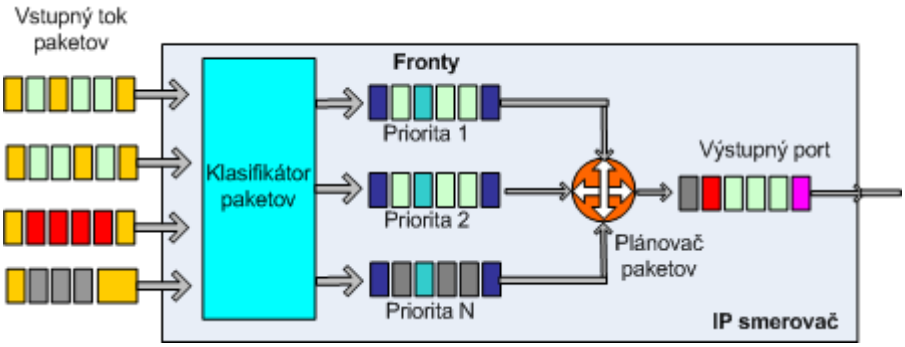
Z toho vyplýva, že hlavným nedostatkom FIFO je, že nerozlišuje triedy prevádzky, v dôsledku toho môžu byť rovnako poškodené všetky toky prevádzky počas preťaženia. Dokonca v tomto prípade FIFO nespracúva značne zmiešanú prevádzku TCP a UDP paketov. FIFO sa v priebehu preťaženia viac prikloňuje k UDP prevádzke pred TCP prevádzkou, z dôvodu potvrdzovania prijímu TCP protokolu čo spôsobuje ešte väčšie preťaženie [2][6].



Obr. 1.1: *Metóda FIFO*

Metóda prioritného radenia do front PQ

Jednoduchý spôsob rozlišovania tried provozu je používanie priority pre radenie do front. V metóde PQ, sa vytvorí N radov s rôznou prioritou od *1* až *N* (obr. 1.2). Plánovanie poradia paketov je vykonané ich priradením do patričnej fronty podľa priority. Pakety z nižších front (napríklad fronta 2) sú spracované len ak už nie sú žiadne pakety vo fronte s vyššou prioritou (napríklad fronta 1). Okamžite ako príde paket s vyššou prioritou, plánovač paketov odloží spracovanie nižších front (fronty 2-N) a spracúva pakety z front s vyššou prioritou (fronta 1).



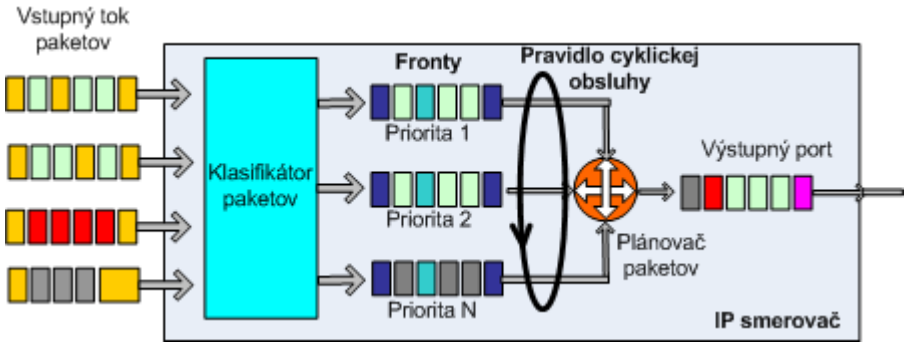
Obr. 1.2: *Metóda PQ*

Tak ako u FIFO tak aj u PQ je hlavnou výhodou jednoduchosť, plus PQ poskytuje jednoduché prostriedky rozlíšenia triedy prevádzky. Hlavným nedostatkom PQ je možnosť vzniku neželaného javu označovaného ako „hladovanie“. Je to odsúvanie radov nižšej priority za paketov s vyššou prioritou. Fronty s nižšou prioritou môžu byť v tomto dôsledku celkom

odrezané od prístupu k výstupného portu. Na túto skutočnosť treba brať ohľad pri aplikácii metódy PQ. Metóda PQ je vhodná, keď prevádzka s vysokou prioritou tvorí len zlomok celkovej prevádzky na všetkých linkách. Je vhodným a jednoduchým prostriedkom radenia do front a poskytnutie požadovanej kvality „real time“ službám, ako je hlas a video. Tieto služby v reálnom čase používajú UDP protokol, ktorý predstavuje nespojový prenos bez potvrdzovania. PQ môže byť použité aj pre TCP prevádzku, ale vyžaduje špeciálne opatrenie, pretože TCP počas preťaženia môže zapríčiniť zväčšenie efektu odsúvania zvyšku prevádzky nižšej priority v dôsledku potvrdzovaného prenosu TCP protokolom. Na predídenie efektu hladovania a zaistenia minimálneho prenosu paketov s nižšou prioritou, môže byť použitá PQ s riadením množstva. V tomto prípade sú pakety s vyššou prioritou zoradené a plánované pred paketmi nižšej priority len ak hodnota prevádzky vo fronte s vyššou prioritou zostáva pod špecifikovanou prahovou úrovňou [2].

Metóda spravodlivého radenia do front FR

Táto metóda sa zakladá na rozdelení prichádzajúcich paketov do N front. Každej fronte je pridelená *1/N* šírka pásma výstupného portu. Pravidelne podľa pravidla cyklickej obsluhy je navštívená každá fronta a prenesený jeden paket, pričom prázdne fronty sú preskočené. Metóda FQ je vo všeobecnosti jednoduchá a to je jej hlavnou výhodou. Šírka pásma je prideľovaná na základe jednoduchého pravidla. Keď je pridaná fronta k existujúcim N frontám pre vytvorenie novej triedy prevádzky, plánovač automaticky prepočíta šírku pásma pridelenú jednej fronte na 1/(N+1) šírky pásma výstupného portu.

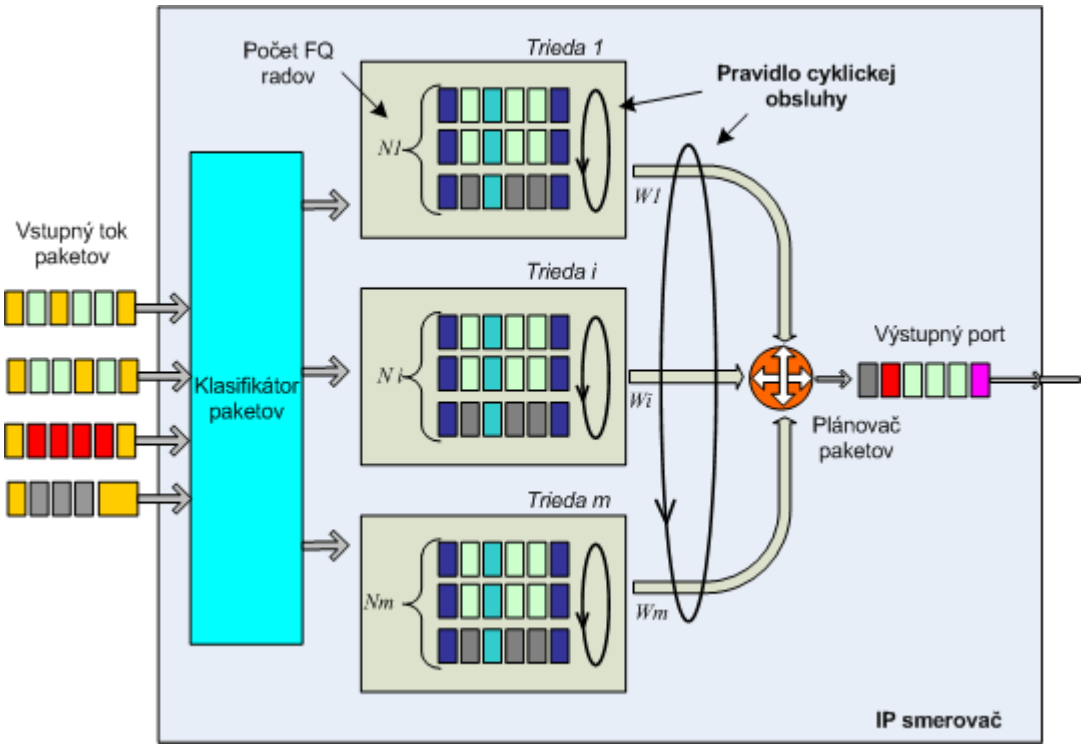


Obr. 1.3: *Metóda FQ*

Metóda FQ má však dve hlavné nevýhody. Každá fronta má pridelenú rovnakú šírku pásma a z toho vyplýva, že nedokáže dostatočne podporiť QoS, kde niektoré služby vyžadujú väčšiu šírku pásma. Druhá nevýhoda súvisí s odoslaním celého paketu pri návšteve, ktorého veľkosť môže byť rôzna a tým sa poruší rovnováha medzi frontami.

Metóda váhovej cyklickej obsluhy WRR

Metóda WRR je tiež označovaná ako metóda radenia do front na základe tried prevádzky CBQ (*Class – Based Queuing*) alebo ako prispôsobené radenie do fronty. Odstraňuje dve hlavné nevýhody metódy FQ. Táto metóda dokáže rozdeľovať šírku pásma výstupného portu podľa požiadaviek tried prevádzky. Vstupné toky sú rozdelené do *m* tried, ktorým je pridelená váha na základe priority a požiadavkou na šírku pásma. Súčet váh priradených jednotlivým triedam musí byť 100% a maximálne využitie linky. V každej triede je vytvorených ďalších *N* front, ktoré sú obsluhované metódou FQ. Pri pridelení šírky pásma triede čo môžeme považovať aj za určitý čas, sú počas tohto času cyklicky prepínané vnútorné fronty tejto triedy. Plánovač strávi nad každou z týchto front rovnaký čas [2].

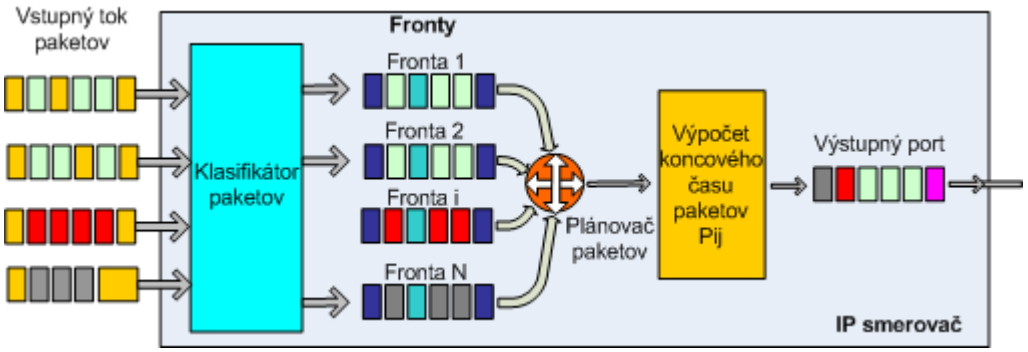


Obr. 1.4: *Metóda WRR*

Metóda váhovej obsluhy spravodlivého radenia do front WFQ

V metóde WFQ je vyriešený problém druhej nevýhodu metódy FQ. Tak ako v FQ sú vstupné toky rozdeľované do *m* radov, ale šírka pásma výstupného portu nie je prerozdeľovaná rovnomerne, ale podľa vhodných váh resp. podľa požiadaviek tried na šírku pásma. Na rozdiel od FQ, kde je vysielaný celý paket, bez ohľadu na jeho veľkosť u metódy WFQ sa počíta koncový čas odoslania paketu a podľa toho sú odosielané. Teoreticky sú zbierané bit po bite z každého paketu a keď je paket celý, je odoslaný na výstup. To zaručí, že

väčšie pakety, musia čakať dlhší čas kým sa poskladajú a budú odoslané na výstupný port. Toto sa však prakticky nepoužíva a v praxi je použitý výpočet koncových časov paketov a podľa toho sú radené a vysielané na výstupný port.



Obr. 1.5: *Metóda WFQ*

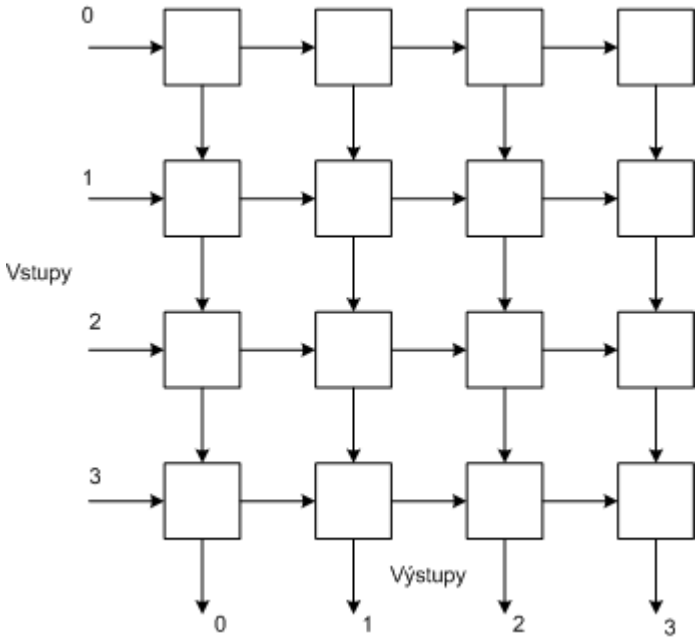
Metóda WFQ založená na triedení prevádzky CB

V tejto metóde rovnako ako v metóde WRR, sú vstupné toky rozdeľované do *m* tried a šírka pásma výstupného portu je prerozdeľovaná týmto triedam podľa vhodných váh určených požiadavkami na šírku pásma. Táto časť (CB) WFQ a WRR je úplne rovnaká. Rozdiel je vo vnútri každej triedy. Rozdiel nastáva pri metóde (CB) WFQ, kde sú aj jednotlivé vnútorné fronty tried plánované metódou WFQ, zatiaľ čo v metóde WRR sú jednotlivé toky plánované metódou FQ.

1.3 Spojovacie pole

Spojovacie pole slúži na fyzické prenesenie paketu zo vstupného portu na požadovaný výstupný port. Druhov spojovacích polí existuje mnoho druhov, pre náš prípad postačí jednocestné spojovacie pole. Jednocestné pole sa vyznačuje hlavne svojou jednoduchosťou. Jednocestné spojovacie polia môžeme rozdeliť do troch hlavných skupín a to na krížové prepínače, na plne prepojené štruktúry a na spojovacie pole typu Banyan. Pre simulovaný model postačí modelovať krížový prepínač. Krížový prepínač (crossbar switch) so štyrmi vstupmi je znázornený na (obr. 1.6). Vodorovné linky odpovedajú vstupným portom a zvislé linky výstupným portom. Sieťový prvok s *N* porty vyžaduje krížový prepínač so štruktúrou *N*N*, obsahujúci N^2 samostatne riadených spínacích prvkov. Každý odpovedá jednej dvojici vstupu a výstupu. Každý spínací prvok má dva stavy: rozopnutý (cross), ktorý je počiatočne nastavený a zopnutý (bar). Spínať každý spínací element je možné úplne automaticky príchodom bunky s odpovedajúcou adresou výstupu. Tento proces môže prebiehať úplne

nezávisle na ostatných bunkách. Toto riešenie riadenia spojovacieho pola výrazne zjednoduší riadenie a spojovacie pole sa stane samosmerovacím. To znamená, že riadiaca funkcia je distribuovaná medzi spínacie elementy. Táto architektúra má radu zaujímavých vlastností. Nehrozí u nej vnútorné blokovanie, majú jednoduchú architektúru a sú modulárne. Najväčšou nevýhodou a obmedzením tohto typu spojovacích polí je, že počet spínacích uzlov narastá kvadraticky. U väčšieho spojovacieho pole sa môže stať úzkym miestom rozhodovanie o výbere bunky pre daný port a časový interval. Aby nedochádzalo k veľkej strate dát a krížový prepínač mohol spracovávať bunky, je nutné ho doplniť o vyrovnávaciu pamäť [6].



Obr. 1.6: *Krížový prepínač*

1.4 Riadiaci člen

Po príchode dátovej jednotky na vstup s požiadavkou na konkrétny výstup je dátová jednotka z pravidla uložená do vyrovnávacej pamäte. Je to z dôvodu, že na jeden výstupný port môže súčasne smerovať niekoľko dátových jednotiek, čo umožňuje spojovacie pole. Iba jedna dátová jednotka môže byť v danom okamžiku odoslaná a ostatné čakajú vo vyrovnávacej pamäti. Prepojovací uzol s podporou QoS musí okrem vyrovnávacej pamäti implementovať aj vhodný algoritmus pre správu front zaisťujúci plánované odoslanie dátových jednotiek, tak aby boli zaistené definované parametre provozu. Môže tiež obsahovať aktívny manažment radenia do front AQM (Active Queue Management). Hlavnou myšlienkou mechanizmu AQM je predchádzať návalom preťaženia a podnikať akcie na predchádzanie alebo zmiernenie účinkov preťaženia a zabránenie celkovej TCP synchronizácii. V skutočnosti sa používajú najmä nasledovné tri metódy AQM:

- Náhodné včasné odhodenie RED (*Random Early Discarding*)
- Váhové náhodné včasné odhodenie WRED *(Weighted Random Early Discarding)*
- Explicitné hlásenie preťaženia ECN (*Explicit Congestion Notification*)

Techniky RED a WRED zahrňujú uskutočňovanie akcií na zahadzovanie paketov vo frontách a nezapájajú priamo koncového užívateľa. ECN využíva zásadne odlišný postup, ktorý vyžaduje priame zapojenie koncového užívateľa. Spomenuté funkcie dátovej trasy sú najčastejšie riešené hardwarovo. Každá z funkcií dátovej trasy by malo byť možné vykonať v čase trvania najkratšej platnej dátovej jednotky. Táto podmienka musí byť splnená aby prepojovací uzol mohol pracovať s plnou rýchlosťou linkového rozhrania. V prípade absencie aktívneho manažmentu je použitá metóda straty časti (*tail drop method*). Je to pasívna technika zahodenia všetkých dátových jednotiek, ktoré sú prijaté v prípade plnej fronty. Jej výhodou je jednoduchosť, ale veľkou nevýhodou je jav nazývaný TCP celková synchronizácia. Pri prenosoch TCP spojení a zahodení niektorých paketov, nastáva negatívne potvrdenie, u ktorého sa predpokladá, že bol stratený v dôsledku preťaženia siete. Vysielač zníži vysielaciu rýchlosť, pre odľahčenie siete. To však spôsobí, že pri zahodení paketov viacerých TCP spojení, dôjde k výraznejšiemu poklesu vyťaženia linky. Následne budú všetky spojenia zvyšovať svoju rýchlosť až do bodu preťaženia, kde budú znova ich pakety zahodené. Dostanú sa do stavu kolísania medzi preťažením a stavom mimo preťaženia nazývaným globálna TCP synchronizácia, spôsobujúci neefektívne využitie kapacity prenosovej cesty [2].

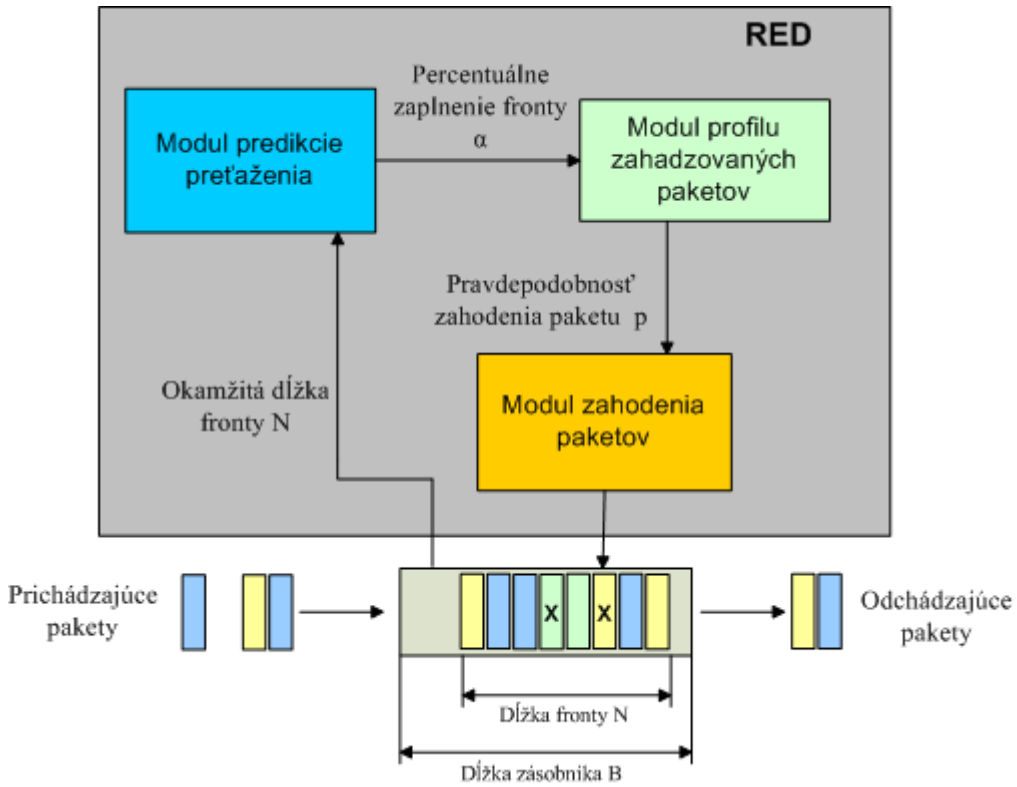
Náhodné včasné odhodenie RED (*Random Early Discarding*)

RED detekuje nárast zahltenia a náhodne zahadzuje pakety z vyrovnávacej pamäti. Vnútorná štruktúra Náhodného včasného odhodenia je znázornená na obrázku 1.7. Zahrňuje algoritmus predikcie zahltenia a profil zahodenia paketu ako centrálnu jednotku.

Hlavná funkcia modulu predikcie zahltenia je odhadnúť ako sa správa prevádzka počas času vo fronte a identifikovať každý nárast zahltenia. Najjednoduchší postup je zistenie aktuálnej dĺžky fronty *N* a určenie zahltenia porovnaním s maximálnou veľkosťou fronty *B*. Dômyselnejšie riešenie zahrňuje výpočet váženého priemeru dĺžky fronty η_N, ktorá je výstupom modulu predikcie zahltenia. Aj keď sa v nej odzrkadľuje aktuálna dĺžka fronty, tak jej presne nezodpovedá. Percentuálne naplnenie fronty je definované ako α :

$$\alpha = \frac{\eta_N}{B} \quad [\%] \tag{1.1}$$

Ďalším modulom v RED je profil zahodenia paketu. Tento modul zodpovedá za vyhodnotenie situácie a prípadné zahodenie paketov. Rozhodnutie o zahodení paketu závisí na použitom profile a na veličine vstupujúcej do modulu, percentuálne naplnenie fronty α. Profil je možné znázorniť pomocou grafu, príklad je uvedený na obrázku 1.8.



Obr. 1.7: *Princíp činnosti metódy RED*

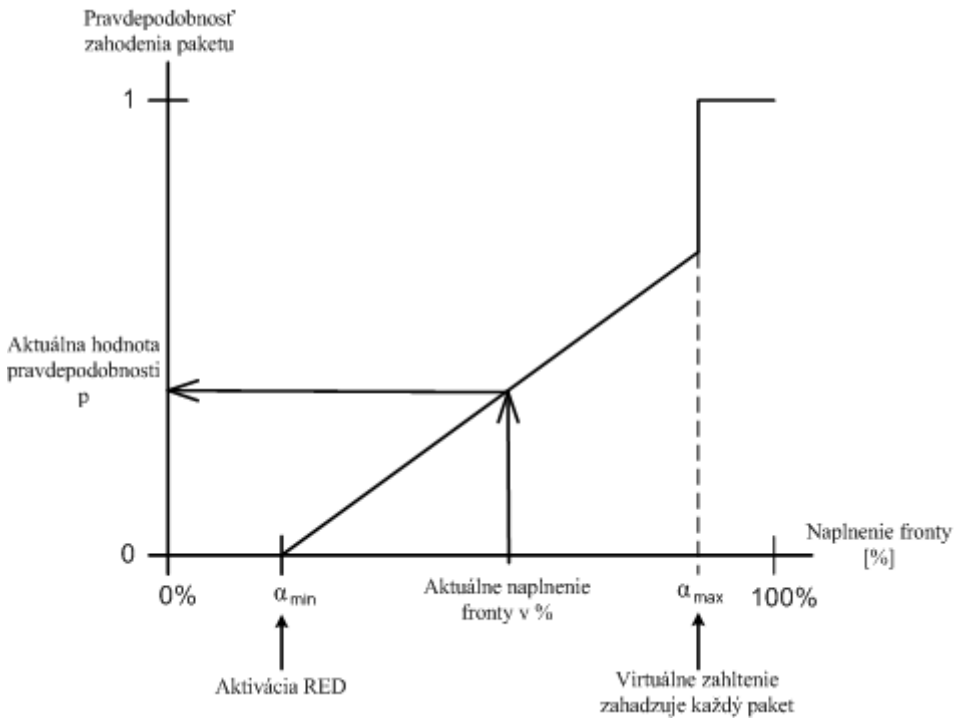
K zahadzovaniu paketov nedochádza pokiaľ α neprekročí prahovú hodnotu α_{MIN}. Po prekročení α_{MIN} sa aktivuje RED, ktorý zahodí paket s pravdepodobnosťou *p*, ktorá je určená na základe použitej funkcie v profile. Na obrázku je znázornená lineárna funkcia, ale môže byť použitá ľubovoľná. Pri prekročení maximálnej hodnoty α_{MAX}, správa fronty zahodí každý

prichádzajúci paket, ale určité miesto vo fyzickej fronte ostáva. Toto opatrenie kompenzuje vlastnosť α, ktorá nezodpovedá aktuálnej veľkosti fronty, ktorá môže byť väčšia. Profil zahodenia paketov je nastavovaný tromi parametrami a to α_{MIN}, α_{MAX} a typom funkcie p=f(α).

Metóda RED sa veľmi nepoužíva, keďže na sieťach prevládajú toky UDP paketov, u ktorých nehrozí globálna TCP synchronizácia a zbytočne by dochádzalo k zahodeniu paketov. Pri použití RED je nutné venovať pozornosť správnemu nastaveniu parametrov profilu zahodenia pre danú sieť, aby nedochádzalo k zbytočnému zahadzovaniu a nedostatočnému využitiu sieťových zdrojov.

Váhové náhodné včasné odhodenie WRED *(Weighted Random Early Discarding)*

Metóda WRED predstavuje použitie RED s viacerými profilmi zahodenia a poskytuje sofistikovanejšiu správu fronty. Pri WRED sú využité pre každú virtuálnu frontu individuálne profily zahodenia. Na viac môžu byť definované viaceré profily v rámci jednej fronty podľa značenia paketov. Podľa tak zvaného farbenia paketov môže byť použitý najviac agresívny profil zahadzovania pre pakety značené červenou a najmenej agresívny pre pakety značené zelenou. Táto vlastnosť umožňuje širokú podporu QoS. Prípustné sú aj rôzne kombinácie profilov zahadzovania paketov. WRED nie je kompatibilné s *Custom Queuing (CQ), Priority Queuing (PQ), Weighted Fair Queuing (WFQ)* a teda nemôže byť použitá na rozhrania využívajúce jednu z týchto metód.



Obr. 1.8: *Profil zahodenia paketov*

Explicitné hlásenie preťaženia ECN (*Explicit Congestion Notification*)

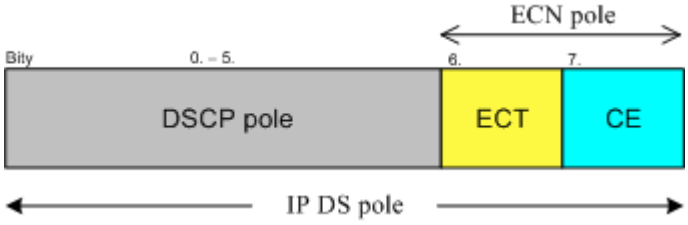
Je to metóda na predchádzanie zahlteniu pre TCP provoz. Bola navrhnutá v roku 1999 v dokumente RFC 2481, ako experimentálny prídavok k IP architektúre. Rozdielny prístup ECN oproti metódam RED a WRED spočíva v informovaní koncových systémov o náraste zahltenia pomocou označenia príslušného poľa v TCP a IP záhlavý. Je uprednostnené indikovanie zahltenia pred zahadzovaním paketov. To znamená, že na rozdiel od RED, kde sú náhodné pakety zahodené, pri ECN sa týmto paketom pridá značka zahltenia na informovanie koncových systémov. ECN pre výber paketov, ktoré budú označené používa rovnaký algoritmus ako RED. Metóda sa dá rozdeliť na tri základné činnosti:

- ECN značkovanie v IP záhlaví
- ECN značkovanie v TCP záhlaví
- ECN nadviazanie spojenia a operácie

ECN vyžaduje vytvorenie oboch záhlaví, IP aj TCP. Využíva zhodne dva rezervované bity v oboch záhlaviach. V prípade IP záhlavia sú to posledné dva bity v poli označujúcom typ služby (*Type of Service* - TOS), pre IPv4 a posledných osem pre pole triedy provozu (*Traffic Class*) v IPv6 záhlaví. Siedmy bit zľava označuje možné použitie ECN (*ECN-Capable Transport*)(ECT) a posledný, teda ôsmy zľava vzniknuté zahltenie (CE). ECT bit je nastavený zdrojovou TCP aplikáciou na jednotku, ktorá indikuje smerovačom v IP sieti, že pakety sú vhodné pre použitie ECN. V prípade očakávaného zahltenia smerovač nastaví bit CE na jednotku, ktorá informuje koncové systémy o zahltení. V prípade príchodu paketu na smerovač s plnou frontou je paket zahodený ako pri metódach RED alebo straty časti.

Podobne ECN definuje dve značky, využívajúce dva bity z rezervovaného poľa TCP záhlavia. Tieto dve značky sú uvedené v RFC 2481, ako ECN-echo a zníženie okna zahltenia (CWR). Umiestnenie týchto značiek v TCP záhlaví a priamo v rezervovanom poli je na obrázku 1.9.

Zdrojové a cieľové TCP aplikácie používajú značky ECN-echo a CWR spolu s bitmi potvrdenia (ACK) a synchronizácie (SYN) z kódového poľa pre nadviazanie spojenia ECN. Po úvodnom nadviazaní spojenia je ECT bit nastavený na jednotku a CE bit na 0. V prípade detekcie nárastu zahltenia smerovačom (algoritmus detekcie zhodný s RED) nastaví CE bit na jednotku. Prímajúca TCP aplikácia zmení ECN-echo bit na jednotku v TCP potvrdzujúcom pakete na informovanie zdroja o vzniknutom zahltení v sieti.



Obr. 1.9: *Umiestnenie značiek ECT a CE v poli TOS*

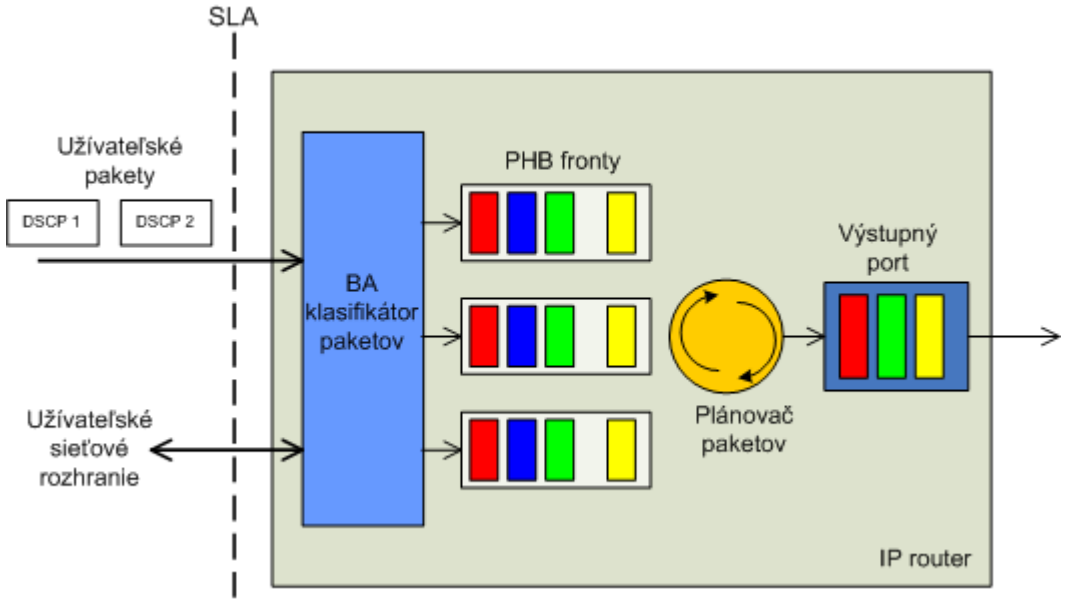
2. Diferencovanie služieb (DiffServ)

Stále sa meniace nároky na siete a vznik citlivých aplikácií, ako VoIP a video konferencie malo za následok vytvorenie požiadavky na technológiu rozlišujúcu druhy dátových tokov. Schopnosť poskytnúť predpovedateľné a rozlíšené úrovne služieb je kľúčové na zaistenie, že so všetkými aplikáciami bude zaobchádzané, tak ako vyžadujú pre svoju správnu činnosť. Je mnoho spôsobov definície QoS. Technická definícia označuje QoS ako sadu techník na správu šírky pásma, oneskorenia, kolísania oneskorenia a stratovosti paketov pre dátové toky v sieti. Účelom každého QoS mechanizmu je ovplyvniť aspoň jeden z týchto štyroch parametrov a v niektorých prípadoch všetky štyri. Diferencovanie služieb sa dá rozdeliť do štyroch celkov:

- Architektúra diferencovaných služieb (DiffServ)
- DiffServ značkovanie paketov
- DiffServ kódové slovo (DSCP)
- Správanie počas preskoku (PHB)

Celkový pohľad na DiffServ

V koncepte DiffServ nie sú jednotlivé dátové toky významné a sú agregované do malej skupiny tried prevádzky. V DiffServ je šírka pásma a ďalšie sieťové zdroje prednostne prideľované agregovaným triedam ako jednotlivým dátovým tokom. Hlavné ťažisko DiffServ je kladené na jednu DS doménu pred celou trasou end-to-end, ktorou prechádza paket. DiffServ poskytuje iba relatívne diferencovanie služieb, samostatne nedokáže poskytnúť absolútnu úroveň QoS. Na okrajoch DS domény je potrebná kontrola prístupu, na kontrolu množstva dát vstupujúcich do siete.



Obr. 2.1: *Príklad konštrukcie Diffserv*

Pojem DiffServ popisuje celkovo zaobchádzanie s dátami zákazníka v sieti poskytovateľa a definuje služby, ktoré môže zákazník očakávať od poskytovateľa. Rozlíšenie služieb je definované v podobe Dohody o úrovni služby (SLA) medzi zákazníkom (napríklad konkrétnou aplikáciou ako VoIP, TCP a ďalšie) a sieťou DiffServ poskytovateľa. Parametre sú vyjadrené v podobe, ktorej zákazník rozumie ako sú Dohoda o prenosových podmienkach (TCA), prevádzkové profily, metriky vlastností (priepustnosť, oneskorenie...), ako je zaobchádzané s nezodpovedajúcim paketom a ďalšie značkovanie a tvarovanie prevádzky. Vzhľadom na DiffServ definíciu služby je na poskytovateľovi navrhnúť jeho sieť tak aby zaobchádzanie s paketmi a správanie očakávané zákazníkom zodpovedalo SLA.

Obrázok 2.1 zobrazuje základné kroky obsiahnuté v poskytovaní DiffServ služieb. Väčšina krokov je internou záležitosťou siete a nie sú priamo viditeľné zákazníkovi. Paket prichádza do routeru s DSCP označením alebo neoznačený. Router skúma DSCP pole paketu a klasifikuje paket do jednej zo skupiny správania (BA). Paket priradený do príslušnej BA je posunutý príslušnému PHB definovanému pre túto BA. Každé PHB je reprezentované DSCP hodnotou a určuje jedinečné zaobchádzanie počas smerovania. A následne sú použité nástroje ako dozor nad prevádzkou, tvarovanie prevádzky, zahadzovanie paketov, aktívna správa front a plánovanie paketov ak je to vhodné.

2.1 Architektúra DiffServ

Na architektúru DiffServ môžeme nahliadať dvomi rôznymi spôsobmi a teda ju rozdeliť na architektúru z pohľadu prenosovej siete a na vnútornú štruktúru DiffServ. V prípade nasadenia v sieti typu IP môžeme uvažovať nasledovne.

Všeobecne môžeme doménu v poňatí IP siete chápať ako geografickú oblasť s hranicami, v rámci ktorých je implementovaná určitá politika alebo technika. IP doména je IP sieť, ktorá je pod kontrolou jednej administrátorskej autority. Môže pozostávať z viacerých sietí, ktoré sú geograficky rozptýlené, ale pod rovnakou autoritou. Sieť je považovaná za DS-schopná, ak je schopná poskytnúť DiffServ. IP doména môže mať časti, ktoré sú DS-schopné a časti bez schopnosti podpory DiffServ. Kľúčové pojmy používané k popisu DS architektúry sú definované v dokumente RFC 2475. Uzly v rámci jednej DS domény sa rozdeľujú na dva druhy a to hraničné uzly a vnútorné uzly. Hraničný uzol môže byť vstupný alebo výstupný, ktoré vykonávajú určité požadované funkcie ako dozor nad prevádzkou. DS doméne sa môžu nachádzať aj uzly nepodporujúce diferencovanie služieb.

Vo všeobecnosti každá individuálna DS doména pracuje so svojou vlastnou politikou a správaním pri preskoku(PHB). Každá DS doména môže používať vlastné kódové slová diferencovania služieb (DSCP) na označovanie druhov prevádzky. Na zaistenie DiffServ v celej sfére DS, musí byť vytvorené rozhranie SLA medzi jednotlivými DS doménami. Časť SLA zaisťuje dohodu autorít na tom ako bude interpretované DSCP, ako bude zaobchádzané s paketami a ako budú predané z jednej DS domény do druhej. Na zaistenie DiffServ v režime end-to-end, musí byť SLA dohodnuté medzi všetkými dotknutými spravujúcimi autoritami.

V prípade zamerania sa na vnútornú architektúru DiffServ je badateľná komplikovaná štruktúru s mnohými komponentmi. Každý z týchto komponentov má v sieti inú úlohu. Hlavné súčasti DiffServ štruktúry vykonávajú nasledujúce úlohy:

- Klasifikácia paketov
- Značkovanie paketov
- Manažment zahltenia
- Predchádzanie zahlteniu
- Úprava prevádzky

Tieto mechanizmy môžu byť implementované samostatne alebo v spolupráci s ostatnými.

Klasifikácia paketov

Môže sa jednať o jednoduchú klasifikáciu na základe informácií druhej alebo tretej vrstvy a je to skupina mechanizmov, ktoré dokážu oddeliť jeden typ paketov od druhého. Príkladom jednoduchého mechanizmu klasifikácie je porovnávanie s tabuľkou (access list), kedy sa porovnáva určitý parameter. O mnoho komplexnejším sú riešenia, ktoré dokážu porovnávať vlastnosti na úrovni štvrtej až siedmej vrstvy. Jedným z tejto skupiny je mechanizmus sieťovo založené rozpoznávanie aplikácií (NBAR) dostupný v routroch od firmy Cisco. NBAR je tiež schopný stavovej inšpekcie paketov, čo výrazne zvyšuje jeho potenciál. Klasifikácia paketov je typicky vykonávaná čo najbližšie k ich zdroju a zvyčajne v kombinácii so značkovaním paketov.

Značkovanie paketov

Značkovanie spočíva v pridávaní značky paketu sieťovým zariadením na základe jeho klasifikácie, takže môže byť jednoduchšie rozpoznaný v nasledujúcich sieťových zariadeniach. Jeden z dôvodov, prečo je model DiffServ taký škálovateľný je, že komplex klasifikácie a značkovania je doporučené na nasadenie iba na prvom zariadení pracujúcom na tretej vrstve ISO. V bežnej podnikovej sieti to znamená, že zariadenie oddelenia, ktoré slúži iba malej časti užívateľov z celej komunity, vykonáva komplexné operácie pre túto vetvu. Potom je značkovanie nesené s paketom cez sieť, čím znižuje záťaž v jadre siete na prepínanie paketov na príslušné rozhranie.

Manažment zahltenia

Celková funkcia manažmentu zahltenia je oddeliť rôzne triedy prevádzky a stanoviť priority prístupu každej triedy k rôznym sieťovým zdrojom. Typicky je manažment zahltenia primárne zameraný na preskladanie paketov pre vysielanie. Toto má dopad na celkovú šírku pásma pridelenú každej triede, ale tiež ovplyvňuje oneskorenie a jitter každej triedy. Manažment zahltenia je využívaný na všetkých sieťových úrovniach (prístup, distribúcia a jadro). Jeho použitie je však dobrovoľné.

Predchádzanie zahlteniu

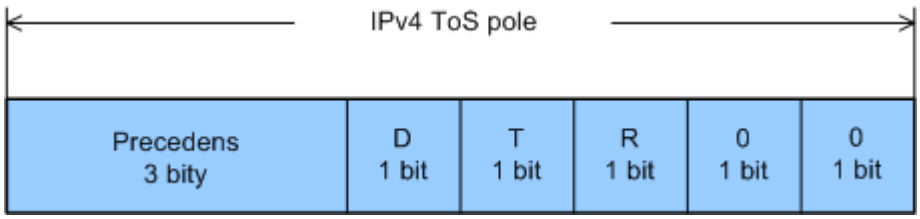
Predchádzanie zahltenia je špeciálne navrhnuté na zahodenie paketov aby sa predišlo stavu zahltenia. Koncept predchádzania zahltenia je založený na prevádzke TCP protokolu. Podrobnejšie je to popísane už v kapitole 1.4, kde sú popísane aj jednotlivé metódy predchádzajúce zahlteniu.

Úprava prevádzky

Úprava prevádzky pozostáva z dvoch hlavných komponentov:

Policer – Policer zabezpečuje politiku prevádzky, čo znamená zahadzovanie paketov, ktoré presiahnu definovanú rýchlosť, s ktorou záťaž prechádza cez policer. Príkladom policerov implementovaných v Cisco sú zviazané prístupové rýchlosti (CAR). Cieľom určenej politiky je klasifikovať limit prevádzky. Príkladom praktického použitia pre policer je limitovanie množstva FTP prevádzky, ktorej je dovolené ísť na výstupné rozhranie na rýchlosť 1 Mbps. Záťaž presahujúca hodnotu 1 Mbps je zahodená. TCP znovu vysielajú zahodené pakety, UDP nie, preto je na zvážení, či je použitie policera pre danú situáciu vhodné.

Shaper – Jeho úlohou je tvarovanie prevádzky. Existuje mnoho druhov tvarovania, prispôsobených presným požiadavkám, ale koncept zostáva rovnaký pre všetky. Cieľom tvarovania je obmedziť rýchlosť, s ktorou prechádzajú cez tvarovací blok ukladaním paketov, ktoré presahujú limit a ich neskorším odoslaním. To je rozpor s princípom policera, kde sú pakety zahadzované ak je prekročená maximálna hodnota. Oboje má svoje výhody a nevýhody, preto musí byť každá situácia hodnotená samostatne, ktorý postup bude výhodnejší. Napríklad služba FTP, ktorá je založená na TCP a je v celku tolerantná k strate paketov, môže prejsť policerom bez negatívneho dopadu na použiteľnosť aplikácie. Pretože FTP neobmedzuje ani oneskorenie paketov pri prenose, môže byť použité aj tvarovanie prevádzky bez dopadu na službu. VoIP spojenie, je na druhú stranu citlivé na oneskorenie a preto je vhodnejšie zahodiť VoIP paket ako ho zdržať v pamäti. Jednotka a spôsob merania je rovnaký ako pri CIR.



Obr. 2.2: *Štruktúra pola ToS*

2.2 Značkovanie paketov

DiffServ používa pole Typu služby (ToS) v záhlaví IPv4 a pole triedy prevádzky (TC) v záhlaví IPv6 pre značkovanie paketov. V prípade, že router pracuje v obvyklom móde a nerozoznáva DiffServ, tak sú polia ToS a TC použité ako boli pôvodne navrhnuté. Keď rovnaký router podporuje DiffServ a pracuje ako DS uzol, tak je ToS a TC pole zrušené a predefinované ako DS pole. Inak povedané, v IP záhlaví nie je definované samostatné pole pre DiffServ.

Značenie paketov v bežnom móde

Pole ToS v záhlaví IPv4 sa skladá z ôsmych bitov, štruktúra pola je zobrazená na obrázku 2.2. V konvenčnom routri nevyužívajúcom DiffServ je týchto osem bitov definovaných podľa RFC 719. Prvé tri bity z pola ToS (0, 1 a 2) sú značené ako IP precedens. Tabuľka 2.1 uvádza jednotlivé významy nastavenia IP precedensu a odpovedajúceho významu. Hodnota prvých troch bitov ´111´ podľa RFC719 je určená len v rámci privátnej siete, kombinácia ´110´ pre medzi sieťovú komunikáciu brán. Nasledujúce tri bity v poli ToS (3, 4 a 5) určujú vlastnosti služby generujúcej pakety. Tabuľka 2.2 uvádza konkrétny význam nastavenia D-bitu, T-bitu a R-bitu z pola ToS. Posledné dva bity (6 a 7) sú vyhradené pre budúce použitie.

V prípade záhlavia IPv6 je obdobne použité pole triedy prevádzky (TC) ako pole ToS v IPv4. V protokole IPv6 je pole TC nasledované poľom Označenie toku (FL), toto pole je tiež relevantné pre QoS, zatiaľ ho však nevyužívajú žiadne dôležité aplikácie.

IP precedence bity	Typ prevádzky	
111	Sieťová kontrola	
110	Medzi sieťová kontrola	
101	Kritické / stav núdze	
100	Flash override	Stupne priority
011	Flash	
010	Immediate	
001	Priority	
000	Routine	

Tab. 2.1 *Významy IP precedensov*

Hodnota bitu	D-bit	T-bit	R-bit
0	Norm. oneskorenie	Norm. priepustnosť	Norm. spoľahlivosť
1	Nízke oneskorenie	Vysoká priepusnosť	Vysoká poľahlivosť

Tab. 2.2 *Význam nastavení príznakových bitov*

2.3 DiffServ kódové slovo

Výraz DSCP označuje prvých šesť bitov poľa ToS alebo TC. Šesť bitov poskytuje šesťdesiatštyri možných permutácií hodnoty poľa DSCP. Týchto šesťdesiatštyri možných hodnôt je rozdelených do troch skupín. (viď. Tab. 2.3)

Posledný bit (šiesty) v skupine jedna je pevne nastavený na nulu. Ostatných päť bitov skupiny jedna môže nadobúdať hodnotu ´0´ alebo ´1´, čo poskytuje tridsaťdva možností pre skupinu jedna DSCP. Skupina jedna vyjadruje štandardné akcie podľa IETF, ktoré sú univerzálne rozpoznávané.

Posledné dva bity skupiny dva sú pevne nastavené na ´11´. Zostávajúce štyri bity dovoľujú 16 permutácií v skupine dva. Skupina dva DSCP nevyžaduje štandardné akcie a je určená pre experimentálne a lokálne účely. DiffServ pakety v súkromnom intranete môžu byť označené aj skupinou dva, majú však iba lokálny význam a nebudú rozoznané v internete.

Skupina tri má vždy posledné bity nastavené na ´01´a ponúka šestnásť možností. Táto skupina je podobná skupine dva, v tom že je určená na experimentálne a lokálne účely, rozdiel je v tom, že skupina tri môže byť použitá pre štandardné akcie ak je to nevyhnutné. V prípade, že je potrebné spraviť DS uzol spätne kompatibilný s klasickým routrom.

Osem kombinácií z DSCP skupiny jedna je použitých na označenie IP precedensov prevádzky konvenčného routera. Týchto osem sa označuje ako konvertor tried kódového slova (CSCP). Posledné tri bity v CSCP sú pevne nastavené na ´000´. Teda CSCP ma podobu ´xxx000´, kde ´x´ môže byť ´0´ alebo ´1´. Keď nie je preddefinované zaobchádzanie s paketom je nastavené danému paketu DS uzlom. DSCP je implicitne ´000000´, čo značí zaobchádzanie najnižšej priority (best effort). V CSCP sú kombinácie ´111000´ a ´110000´ učené na špeciálne účely. Korešpondujú s IP precedensom ´111´ a ´110´, ktoré sú bežne používané pre sieťovú kontrolu a medzi sieťovú signalizáciu v bežných uzloch.

Skupina (Pool)	Kódové slovo	Účely
1	xxxxx0	Štandardné akcie
2	xxxx11	Experimentálne / lokáne
3	xxxx01	Experimentálne / lokáne*

* tieto hodnoty môžu byť použité pre budúce štandardy. 'x' predstavuje 0 alebo 1

Tab. 2.3 *Skupiny DSCP*

2.4 Správanie počas preskoku (PHB)

DiffServ využíva klasifikačnú metódu Súhrnného správania (BA). V BA klasifikačnej metóde sú pakety klasifikované na základe DSCP hodnoty ako jediného parametra. S paketmi zaradenými do BA je zaobchádzané rovnako.

Ako je v routri s paketmi zaobchádzané je externe pozorovateľné, aj keď je to záležitosť vnútornej implementácie v rámci routra. PHB je externe pozorovateľné v tom zmysle, že definuje správanie pozorované mimo routra. PHB je technický popis vnútornej siete a nie je pozorovateľný pre koncového užívateľa.

DiffServ je zásadne preskokovo orientovaný a nie je postavený na definovaní PHB pre jednotlivý sieťový prvok. Ale na poskytnutie DiffServ naprieč celou DS doménou, musí byť PHB pre jednotlivé routre navrhnuté tak, aby bola zaistená celková end-to-end QoS čo najlepšie. Správne navrhnuté PHB kombinované s vhodnou úpravou prevádzky, vstupnou kontrolou a zaručením sieťových služieb dokáže zabezpečiť DiffServ pre end-to-end spojenia. Poskytovatelia sieťových služieb používajú PHB na určenie šírky pásma a ostatných sieťových zdrojov. V PHB je možné definovať požiadavky na sieťové zdroje (šírka pásma, veľkosť fronty...), relatívnu prioritu medzi viacerými PHB v jednom sieťovom prvku a očakávané charakteristiky pre oneskorenie, zmenu oneskorenia a stratovosť paketov.

Na vytvorenie zaobchádzania s posielanými paketmi definovaného PHB musí interný mechanizmus routra implementovať vhodný aktívny manažment front, techniku plánovanie paketov a ostatné požiadavky diskutované v tejto kapitole. PHB routra neurčuje použité vnútorné mechanizmy, existuje mnoho možných riešení. Preto implementácia PHB je vlastným riešením každého výrobcu sieťových prvkov a nie je všeobecne štandardizovaná. Existujú však dva typy štandardov PHB: Urýchlené odoslanie (EF) a zaručené odoslanie (AF).

Urýchlené odoslanie (EF)

Urýchlené odoslanie (expedited forwarding) PHB bolo prvotne špecifikované v dokumente RFC 2598, ktorý bol neskôr nahradený RFC 3246. Doporučená DSCP hodnota pre EF PHB je ´101110´. Pri použití EF sú pakety smerované s nízkou stratovosťou, malým oneskorením a malou zmenou oneskorenia. Pre dosiahnutie týchto vlastností vyžaduje vyhradenie určitej šírky pásma linky výstupného portu. Správna funkcia EF je možná, ak šírka pásma výstupnej linky plus veľkosť fronty a ostatné sieťové zdroje využívané EF paketmi dosiahnu servisnú rýchlosť μ, ktorá bude vyššia ako množstvo prichádzajúcich paketov λ.

Vhodné použitie EF je pre emuláciu okruhov, emuláciu privátnych liniek a služby reálneho času ako hlas a video, ktoré sú náchylné na stratovosť, oneskorenie a kolísanie oneskorenia.

Na zaistenie EF PHB naprieč celou DS doménou, musí byť nastavená správna šírka pásma výstupných portov na všetkých vnútorných routroch aby bola splnená podmienka $\mu > \lambda$.

Pokiaľ zariadenie pracuje v rámci podmienky $\mu > \lambda$, môžu byť navštívené aj fronty nižšej priority a tým sa predíde hladovaniu neprioritných front. Tento druh fronty sa tiež

nazýva rýchlostne obmedzená prioritná fronta. Alternatívne možnosti sú použitie striktného mechanizmu plánovania odosielania, kde musí byť kladený veľký dôraz na to aby sa predišlo hladovaniu. Druhov alternatívou je implementácia EF PHB využívajúcu niektorú variantu WFQ.

Odkedy je EF primárne používané pre služby reálneho času a odkedy tieto služby využívajú UDP protokol namiesto TCP, je technológia RED nevhodná pre fronty EF. Nevhodnosť spočíva v neschopnosti aplikácií využívajúcich UDP reagovať na zahodenie náhodného paketu. V prípade ak nie je použitý vhodný AQM pre EF PHB, je najlepšie použiť metódu tail drop.

Zaručené odoslanie (AF)

Zaručené odoslanie (AF) PHB je definované dokumentom RFC 2597. Zameranie tejto metódy je na doručenie paketov spoľahlivo a preto oneskorenie a kolísanie oneskorenia nie sú až tak dôležité ako stratovosť paketov. Je to vhodné hlavne pre služby nevyžadujúce spojenie v reálnom čase, teda pre TCP aplikácie. AF je odvodený z všeobecnej triedy aktívneho manažmentu front označovaného RIO (RED with the In/Out bit). Avšak PHB nedefinuje žiadnu presnú implementáciu do routrov a AF PHB nedefinuje RIO. Jednoducho to popisuje ako zaobchádzanie pri posielaní charakteristické pre RIO. Nastavenie správania AF konkrétneho vyhotovenia je znovu na prevádzkovateľovi siete. Základný koncept RIO rozdeľuje pakety na dve skupiny zhodné (in-profile) a nezhodné (out-of-profile), podľa toho ako zodpovedajú pravidlám politiky a prislúchajúco ich značkuje. RED v routri využíva tieto značky pri výbere paketu na zahodenie, pakety zodpovedajúce profilu majú nižšiu pravdepodobnosť zahodenia.

AF je založené na rozšírení binárneho značkovania RIO až na tri úrovne pravdepodobnosti zahodenia. Ako prvé sú nastavené štyri triedy posielania (AF1,AF2, ...AF4). V rámci každej triedy sú definované tri podtriedy s vlastnou pravdepodobnosťou zahodenia. V tabuľke 2.4 sú uvedené všetky AF triedy aj s 12 podtriedami a ich DSCP hodnotami podľa RFC 2597. Dokument taktiež dovoľuje pridanie ďalších úrovní pravdepodobnosti zahodenia, ale s významom len pre lokálne účely.

PHB trieda	PHB podtrieda	Drop precedens	DSCP
	AF 41	Nízky	100010
AF 4	AF 42	Stredný	100100
	AF 43	Vysoký	100110
	AF 31	Nízky	011010
AF 3	AF 32	Stredný	011100
	AF 33	Vysoký	011110
	AF 21	Nízky	010010
AF 2	AF 22	Stredný	010100
	AF 23	Vysoký	010110
	AF 11	Nízky	001010
AF 1	AF 12	Stredný	001100
	AF 13	Vysoký	001110

Tab. 2.4 *Rozdelenie a popis tried AF*

V prípade, že AF prevádzka na vstupom porte presiahne prednastavenú rýchlosť, nezhodné pakety nebudú doručované s tak vysokou pravdepodobnosťou ako zhodné pakety. V prípade zahltenia siete sú nezhodné pakety zahadzované skôr ako zhodné pakety. RFC 2597 neupresňuje pre aký typ služieb by malo byť AF použité. Poskytovatelia služieb ale s obľubou využívajú AF vďaka mechanizmom, ktoré umožňujú vytvorenie služby DiffServ s rôznou úrovňou kvality. Jedným z príkladov je tvz. "Olympijská služba", kedy sú pakety rozdeľované do troch skupín služieb, zlatá, strieborná a bronzová využitím troch tried AF. Po navrhnutí rozvrstvenia tried služieb sú kvantitatívne a kvalitatívne rozdiely medzi AF triedami realizované pomocou priradenia šírky pásma, veľkosti vyrovnávacej pamäte a ďalších parametrov. Fyzicky môžu byť realizované tri triedy AF, ako samostatné fronty. Šírka pásma výstupného portu je potom zdieľaná týmito tromi frontami. V rámci každej fronty AF sú pakety značené troma "farbami" vyjadrujúcimi profil pravdepodobnosti zahodenia. Z povahy prevádzky je vhodné použiť kombináciu s niektorým aktívnym manažmentom front.

3. Neurónové siete a možnosti ich použitia v sieťovom prvku

Neurónová sieť je výpočtový model, zostavený na základe abstrakcie vlastností biologických nervových systémov. Základnou časťou neurónovej siete je model neurónu s N vstupmi a M výstupmi, ktorý spracúva informáciu. Neurónovú sieť môžeme považovať za masívny paralelný procesor , ktorý má sklon k uchovávaniu experimentálnych znalostí a ich ďalšieho využívania. Napodobňuje ľudský mozog v dvoch aspektoch, poznatky sú zbierané počas učenia a medzi neurónové spojenia sú využívané na ukladanie znalostí. Medzi výhodné vlastnosti neurónových sietí patrí práve schopnosť učiť sa a teda prispôsobiť situácií. Ďalšou výhodou je schopnosť generalizovať, čo znamená, že sieť dokáže reagovať a správne zaradiť aj vstupné vzory, ktoré neboli súčasťou učenej množiny. Veľkú výhodu to poskytuje neurónovým sieťam pre aplikácie riešiace problémy s obrovským množstvom kombinácií vstupných a výstupných hodnôt. Tieto problémy nie je možné kompletne obsiahnuť do trénovacej množiny. Neurónové siete majú aj svoje nevýhody a medzi ne patria: náročná voľba vhodnej štruktúry siete, zložité a niekedy nemožné natrénovať sieť, náročný odhad výkonnosti budúcej siete, veľkosť a zložitosť. Ich využitie je možné v mnohých oblastiach, ako napríklad: spracovanie obrazu a počítačové videnie, rozpoznávanie znakov a reči, medicína, vojenské systémy a mnoho ďalších.

3.1 Model neurónu

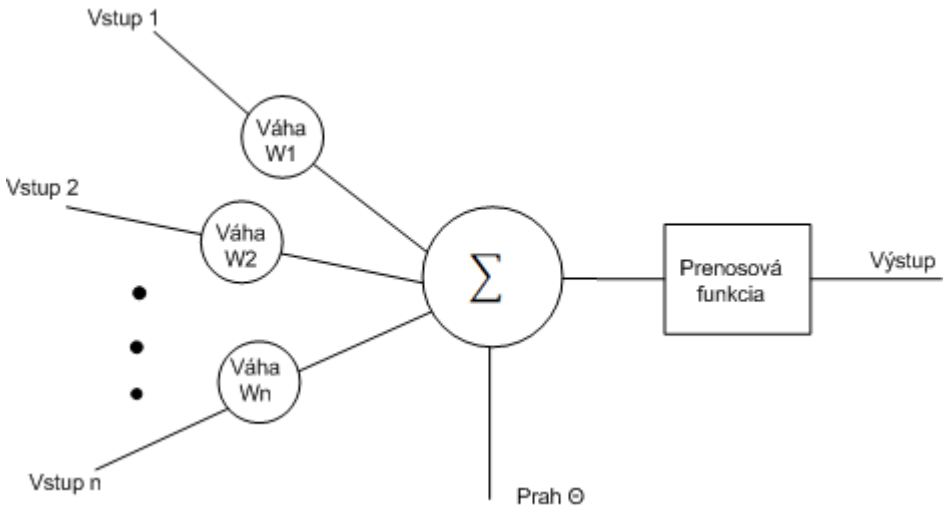
Základným prvkom umelých neurónových sietí je matematický model neurónu. Tento model je vytvorení na základe biologického neurónu, ale jeho cieľom je dosiahnutie najväčšej možnej efektivity, čo neplatí pre presné popisy neurónu. Jednotlivé modely sa líšia použitím vnútornej prenosovej funkcie a štruktúrou. Jeden z prvých modelov je McCulloch-Pittsov model neurónu. Jeho následné vylepšenie je perceptrón (Obr.3.1), ktorý je doplnený o synaptické váhy a prah Θ. Každý z N vstupov neurónu je násobený synaptickou váhou w_{ij}, ktorá je vyjadrením schopnosti neurónu učiť sa. Hodnota váh je hľadaná počas učenia, aby hodnoty vstupov z trénovacej množiny odpovedali výstupom. Aktivačný potenciál u určuje či je neurón aktívny. Ten je pre každý model a sieť odlišný. Pre perceptrón je aktivačný potenciál počítaný podľa rovnice (3.1) [5].

$$u(j) = \sum_{j=1}^{n} w_{ij}\, x_i + \theta_j \tag{3.1}$$

Neurón môže obsahovať aj aktivačnú (prenosovú) funkciu, ktorá ovplyvňuje výstup neurónu. Najbežnejšie funkcie sú:

- Sigmoida $f(u) = \frac{1}{1 + e^{u/t}}$

- Skoková funkcia $f(u) = <_{-1}^{1} {}^{u \geq 0}_{u \leq 0}$
- Lineárna funkcia $f(u) = au + b$
- Hyperbolická tangenta $f(u) = tanh\left(\frac{u}{T}\right)$
- Gaussaova (RBF) $f(u) = e^{-(u^2)}$



Obr. 3.1: *Principiálne zobrazenie perceptrónu*

3.2 Druhy sietí a topológií

Umelé neurónové siete je možné rozdeľovať podľa viacerých kritérií a to podľa:

- **<u>Podľa smeru šírenia informácie</u>:**
  - v priamom smere/dopredné (*feedforward*): od vstupu k výstupu
  - v spätnom smere (*backpropagation*): na princípe spätnej väzby medzi neurónmi

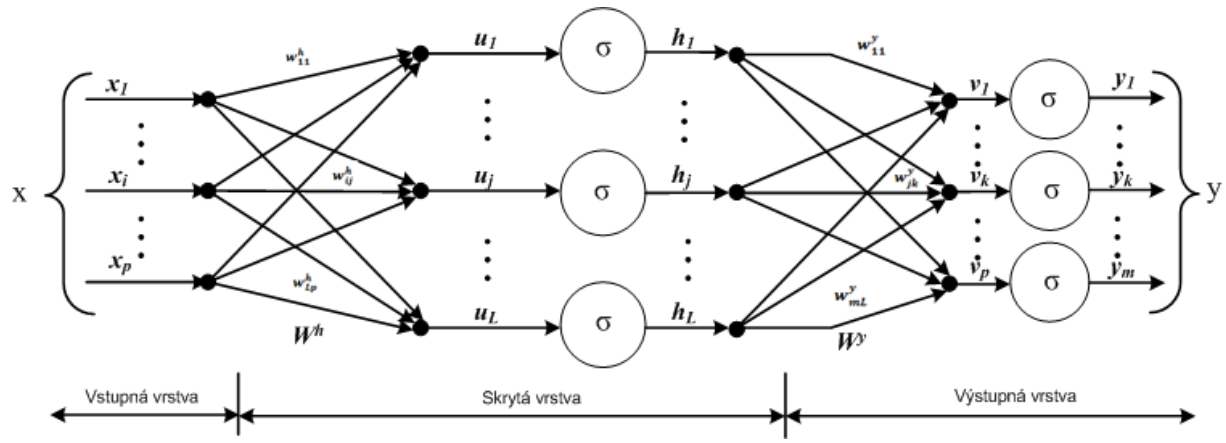
- **<u>Podľa počtu vrstiev, topológie</u>**:
  - jednovrstvové siete: Hopfieldova sieť, Kohenenova sieť...
  - viacvrstvové siete: tvorené min. troma vrstvami, vstupná vrstva sa často neuvádza
  - rekurentné siete: viacvrstvové siete so spätnoväzobnými slučkami
  - mrežová štruktúra: samo organizačné mapy. Jedno, dvoj a viac dimenzionálne polia

- **<u>Podľa algoritmu učenia</u>:**
  - Hebbov zákon učenia: vplyvom aktivity neurónov sa posilňuje, oslabuje ich väzba
  - Zákon kompozice: váhy sa upravujú víťaznému neurónu, prípadne aj jeho okoliu

- Asociatívne: získanie vzťahov medzi vzormi alebo skupinami vzorov
- Učenie s učiteľom: opakovane zadávané vstupné hodnoty a k nim známe výstupy, algoritmus porovnáva výstup a požadovaný známy výstup.
- Učenie bez učiteľa: sieť má schopnosť rozoznať podobné vlastnosti u vstupných vzorov a sústreďuje ich vo svojej štruktúre do oblastí či skupín

3.3 Siete typu Perceptrón

Siete tohto typu sa skladajú minimálne z troch vrstiev. Vstupná vrstva siete slúži ako distribučná, jej úlohou je zmeniť rozmer vstupného vektoru, štandardizuje hodnotu vstupných premenných do rozsahu od -1 do 1. Druhá vrstva je najčastejšie skrytou vrstvou, každý z jej neurónov je spojený s jednotlivými prvkami vstupnej vrstvy. Každá prichádzajúca hodnota od vstupných neurónov je násobená váhou w_{ij} a tieto hodnoty sú združené do kombinovanej hodnoty u_j. Vážená suma (u_j) transformovaná prenosovou funkciou σ, ktorej výstupom je hodnota h_j. Táto výstupná hodnota je distribuovaná do tretej, výstupnej vrstvy. Obe tieto vrstvy majú pevne pripojené váhy a preto sú pre ne konštantné. Tretia a najdôležitejšia vrstva je zložená z perceptrónov, označovaných v tomto prípade aj ako rozpoznávače príznakov. Tieto perceptróny majú na rozdiel od predchádzajúcich vrstiev nastaviteľné synaptické váhy, ktoré sa nastavujú pri učení siete. S hodnotami prichádzajúcimi zo skrytej vrstvy sa vykonávajú totožné operácie ako pri prepojení vstupnej a skrytej vrstvy, čo je dobre viditeľné na obrázku 3.2. Výsledná hodnota y_j je výstupom siete. V prípade, že je analýza vykonávaná so spojitými cieľovými hodnotami, tak vo výstupnej vrstve bude iba jeden neurón. Produkujúci jednu výstupnú hodnotu y. Pre klasifikáciu problému s viacerými kategóriami vstupných premenných je vo výstupnej vrstve N neurónov produkujúcich N hodnôt, pre každú z kategórií vstupných premenných [3].



Obr. 3.2: *Trojvrstvová sieť perceptrónov*

Ukážka siete typu perceptrón zobrazená na obrázku vyššie, je plne prepojená, trojvrstvová sieť s do predným šírením. Plne spojená znamená, že výstup z neurónu je pripojený na každý jeden neurón v nasledujúcej vrstve. Siete tohto typu môžu mať aj viac skrytých vrstiev ako jednu.

Trénovanie a návrh viacvrstvovej siete perceptrónov:

Cieľom trénovacieho procesu je nájdenie vhodnej hodnoty váh, ktoré spôsobia, že výstupné hodnoty neurónovej siete budú čo najbližšie ako je to možné k požadovaným cieľovým hodnotám. Existuje niekoľko otázok, ktoré sú zapojené do návrhu a trénovania viacvrstvovej perceptrónovej siete:

- Rozhodnutie koľko skrytých vrstiev v sieti použiť
- Rozhodnutie koľko neurónov použiť v každej skrytej vrstve
- Nájdenie globálneho optimálneho riešenia, ktoré zabráni vzniku lokálnych miním
- Zaistenie konvergencie k optimálnemu riešeniu v prijateľnom čase

Samotné trénovanie siete sa dá zhrnúť do niekoľkých bodov, v prípade, že máme k dispozícii trénovaciu množinu v tvare $A_{train} = \{(x^1, d_1), (x^2, d_2), \dots, (x^n, d_n)\}$. Množina je zložená z *N* dvojíc vstupných vektorov a im prislúchajúcich výstupov.

Krok 1: Zvolíme rýchlosť učenia $\alpha \in (0, 1\rangle$. Počiatočné váhy (v rátane prahu) inicializujeme na hodnote $\in (-0{,}5; 0{,}5)$. Počítadlá nastavíme na $k = 1$, $n = 1$, kde k je poradové číslo prechodu cez A_{train} a n je index vzoru. Kumulatívna chyba $E = 0$.

$$E = \sum_{n=1}^{n} E_n \tag{3.2}$$

E_n sa bude počítať podľa vzťahu (3.3),v ktorom d je požadovaný výstup a o výstup siete. Zvolíme malé kladné číslo ε, ktoré nám bude slúžiť na zastavenie učenia.

$$E_n = \frac{1}{2}\sum_{k=1}^{K}(d_{nk} - o_{nk})^2 \tag{3.3}$$

Krok 2: Na vstup dáme vzor x^n a vypočítame výstup y^n a o^n

Krok 3: Pre každý výstupný neurón vypočítame δ_{ok} podľa rovnice (3.4) a pre každý skrytý neurón δ_{yj} podľa (3.5). f'_k je derivácia prenosovej funkcie (sigmoida).

$$\delta_{ok} = -\frac{\partial E_n}{\partial(net_k)} = -\frac{\partial E_n}{\partial o_k}\frac{\partial o_k}{\partial(net_k)} = (d_{nk} - o_{nk})f'_k \tag{3.4}$$

$$\delta_{yj} = \left(\sum_{k=1}^{K} \delta_{ok} w_{ok} \right) (f_k)' \tag{3.5}$$

Krok 4: Modifikujeme váhy pre výstupné neuróny $w_{kj} \leftarrow w_{kj} + \alpha \delta_{ok} y_j$ a pre skryté neuróny $v_{ji} \leftarrow v_{ji} + \alpha \delta_{yj} x_i$.

Krok 5: Ak $n < N$ inkrementuj $n = n + 1$ a choď na krok 2, inak choď na krok 6.

Krok 6: Bez modifikácie váh prejdeme cez A_{train} a vypočítame kumulovanú chybu *E*, ktorá nám povie aká je chyba po jednom kole učenia. Ak $E < \varepsilon$, ukončí učenie. Inak: premiešaj vzory v množine A_{train} tak aby prichádzali v každom kole učenia v inom náhodnom poradí. Polož *E=0, n=1, k=k+1* a choď na krok 2. Začína sa nový trénovací cyklus, nové kolo trénovania, čo znamená nový prechod cez A_{train}.

3.4 Hopfieldova sieť

John Hopfield zistil, že použitie energetickej funkcie siete, má veľký význam na správne fungovanie neurónovej siete. Hopfieldova sieť môže slúžiť na optimalizáciu, ako klasifikátor alebo ako autoasociatívna pamäť. Najčastejšie sa používa práve ako autoasociatívna pamäť pri rekonštrukcii poškodených obrázkov. Je to jednovrstvová sieť, ktorá ma rovnaký počet vstupov aj výstupov. Každý neurón má ešte ďalší *N* výstupov, ktoré sú privedené na vstupy ostatných neurónov. Ich binárna aktivita (0,1) je určená princípom, že ak vážená suma vstupných aktivít je väčšia alebo rovná prahovému koeficientu, aktivita daného neurónu je jednotková. V opačnom prípade, ak vážená suma vstupných aktivít je menšia ako prahový koeficient, potom aktivita je nulová. Jednotlivé synaptické váhy medzi dvojcou neurónov sú zhodné a preto ich môžeme reprezentovať ako symetrickú, štvorcovú maticu s nulovou diagonálou. Nuly na diagonále vyjadrujú, že neurón nie je pripojený sám na seba a preto je váha nulová. Hopfield zaviedol namiesto chybovej funkcie, energetickú funkciu siete, ktorá je definovaná pre každý stav siete:

$$E = -\frac{1}{2} \sum_i \sum_j w_{ij} x_i x_j + \sum_i x_i \theta_i \tag{3.6}$$

Hopfieldova sieť dokáže pracovať s bipolárnym (-1,1) aj binárnym (0,1) signálom na vstupe. Medzi pomerne veľké nevýhody tejto siete patrí pamäťová náročnosť, ktorá sa kvadraticky zväčšuje s pridaním vstupu a tým aj jej výpočtové nároky. Nevýhodou je aj malý

počet vzorov, ktorý dokáže uchovať a to iba 0,15 vzoru na neurón. Toto malé číslo, čiastočne kompenzuje fakt, že sieť vďaka svojej štruktúre dokáže rozpoznať aj inverzné vzory [1][5].

Trénovanie Hopfieldovej siete:

Pred začatím učenia siete si musíme vytvoriť *M* matíc vzorov, ktoré budú mať rozmer *NxN*. Jednotlivé matice získame vynásobením vstupov dvoch neurónov a to pre všetky možné kombinácie. Výsledná matica váh je výsledkom súčtu dielčích matíc vzorov. Algoritmus učenia je jednorázový, čo znamená, že keď sieť prejde celú množinu vstupných vzorov je naučená.

Po naučení siete je možné pristúpiť k vybavovaniu, ktoré prebieha iteratívne. Na vstup sú postupne privedené všetky neznáme vzory, z ktorý sa postupne vypočítajú výstupy. Tieto výstupy v nasledujúcej fáze sú použité ako vstupy pre ostatné neuróny. Po každej fáze sa prepočítajú váhy. Tento postup sa opakuje až do chvíle, kým sa výstup neurónu nezmení v priebehu dvoch fáz, potom môže byť tento vzor považovaný, že odpovedá tomuto výstupu.

3.5 Siete typu RBF

Ich názov je odvodený od názvu typu funkcie, ktorá je použitá v neurónoch skrytej vrstvy (Radiálna bázová funkcia). Sieť RBF má pevný počet vrstiev s podobnou štruktúrou ako samo-organizačné siete. Je to dopredná sieť so skrytou vrstvou. Neuróny v prvej vrstve majú pevne nastavené váhy, v druhej vrstve prebieha nastavovanie synaptických váh obdobne ako vo viacvrstvovej perceptrónovej sieti. Neuróny v skrytej vrstve obsahujú Gaussovskú prenosovú funkciu, ktorej výstupy sú inverzné úmerne od vzdialenosti od centra neurónu. RBF siete sa najlepšie používajú v oblastiach klasifikácie a regresie. Princíp fungovania siete by sa dal zhrnúť ako hľadanie najbližšieho neurónu k vzoru a odstupňovanie váh neurónov podľa ich vzdialenosti. Tým je vyjadrený vplyv daného neurónu na predikovaný výsledok. Výsledok môže byť ovplyvnený počtom najbližších susedov, ktorí sa berú do úvahy. Z toho vyplýva nepresnosť RBF sietí, keď sa jednotlivé zhluky prekrývajú.

Trénovanie a rozloženie RBF sietí:

Pre učenie je možné použiť viacero metód. Jedna z metód podobná metóde spätného šírenia chyby v prvom kroku určuje pozíciu stredu zhlukov (*clusters*), ktoré sú následne použité ako stredy RBF funkcií. Ďalšou možnosťou je použiť náhodnú množinu trénovacích bodov ako stredy zhlukov. Tretia možnosť rozmiestnia je stredy zhlukov rozložiť rovnomerne vo vstupnom priestore, toto rozmiestnenie zrýchli učenie a dosahujú sa lepšie výsledky pre vzor neobsiahnutý v učení. Pozícia stredov je vyjadrená hodnotou synaptických váh *wij* medzi vstupnou a skrytou vrstvou. V nasledujúcej fáze prebehne učenie s učiteľom, ktoré modifikuje váhy medzi skrytou a výstupnou vrstvou [4].

3.6 Kohenenove mapy

Kohenenova sieť je typickým predstaviteľom samo-organizujúcich sa neurónových sietí. Učenie prebieha bez učiteľa a preto sieť vykonáva len analýzu vstupných dát a to presnejšie druh zhlukovej analýzy. Sieť predstavuje jedna vrstva radiálnych neurónov, ktoré sú usporiadané do mriežky. Z predstavy mriežky vyplýva skutočnosť, že každý neurón je spojený so svojimi susednými neurónmi. Neurónová sieť má rovnaký počet vstupov aj výstupov. Učiaci mechanizmus musel byť prispôsobený a vznikla metóda súťaživého učenia. Neuróny medzi sebou súťažia a iba jeden môže byť aktívny v danom čase. Poľa chovania okolia rozoznávame stavy excitácie a inhibície. Pri excitácii sa hodnota vnútorného potenciálu okolitých neurónov zvyšuje. Pri inhibícii naopak hodnota vnútorného potenciálu klesá. Neuróny neobsahujú prenosovú funkciu a ich poloha je definovaná hodnotou váh vo vstupnej vrstve. Víťazný neurón sa vyberie na základe prepočtu, kedy sa určí, ktorý z neurónov je najbližšie vstupnému vzoru. Všetky dôležité informácie sú teda uložené vo váhach a jednotlivých neurónov. Vzdialenosť sa vypočítava podľa nasledujúceho vzťahu:

$$d_j = \sum_{i=0}^{N-1} \left[x_i(t) - w_{ij}(t)\right]^2 \tag{3.7}$$

Trénovanie a vybavovanie Kohenenovej siete:

Trénovanie spočíva v usporiadávaní náhodného rozloženia neurónov do mriežky, ktorá je schopná klasifikovať všetky vstupy. Učenie siete prebieha iteratívne a teda krok po kroku, váhy a vnútorný potenciál neurónu sa upravujú v každom kroku učenia a to je vždy pri vstupe vzoru. Trénovací proces by sa dal zhrnúť do nasledovných bodov.[4]

- Vstupné váhy pre všetky spojenia neurónov a vstupov sa nastavia na náhodnú malú hodnotu. Parameter učenia je nastavený na hodnotu blížiacu sa 1. Nasleduje nastavenie hodnôt okolia pre každý vstupný neurón, tak aby okolie pokrývalo všetky neuróny. Ako posledné sa nastaví minimálna hodnota okolia, pod ktorú nesmie klesnúť znižovaním.
- Na vstup neurónovej siete sa predloží nový trénovací vzor
- Prebehne výpočet vzdialenosti d_j medzi vzorom a všetkými výstupnými neurónmi podľa rovnice (3.7)
- Výber najbližšieho neurónu podľa $d_j = min(d_j)$
- Prispôsobenie váhy pre daný neurón a jeho okolie
- Privedenie nasledujúceho trénovacieho vzoru alebo ukončenie učenia po vyčerpaní všetkých vzorov

4. Návrh modelu sieťového prvku v prostredí Matlab/Simulink

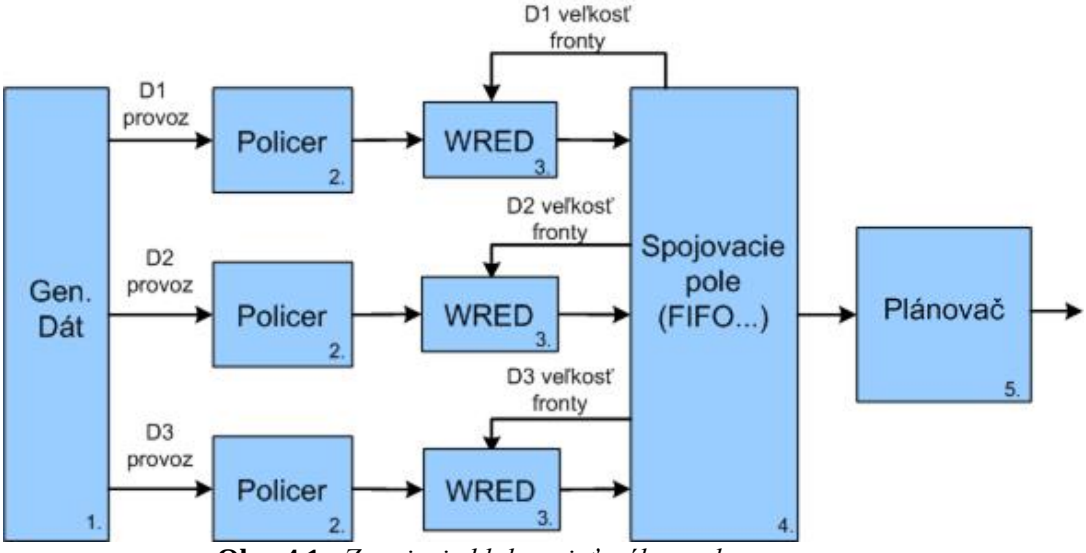
Programy z rodiny Matlab poskytujú programovací jazyk vysokej úrovne, interaktívne technické výpočtové prostredie a funkcie na vývoj algoritmov, analýzu dát a vizualizáciu, numerické počítanie. Vďaka univerzálnosti prostredia a širokým možnostiam použitia vznikli pre Matlab knižnice funkcií, ktoré sú nazývané aj toolboxi. Knižnice rozširujú a uľahčujú použitie programu v ďalších technických a vedných oblastiach. Matlab slúži ako základ a zdroj pre všetky ostatné produkty spoločnosti MathWorks. Medzi najväčšiu a najvýznamnejšiu nadstavbu patrí Simulink. Jeho hlavným zameraním je simulácia dynamických systémov, ktoré sa vytvárajú v podobe blokových schém. Simulink obsahuje interaktívne grafické rozhranie a modifikovateľnú sadu knižníc, ktoré obsahujú bloky. Pomocou blokov je možné navrhnúť, simulovať a testovať množstvo časovo premenných systémov, ako sú komunikácia, spracovanie signálov a mnoho ďalších.

4.1 Konštrukcia sieťového prvku

Realizáciu sieťového prvku ako celku je potrebné rozdeliť na podrobnejšiu štruktúru pre zjednodušenie predstavy. Najsprávnejšie je rozdelenie na logické bloky, ktoré čiastočne odpovedajú teoretickému modelu a každý vykonáva jednu z funkcií. V jednotlivých blokoch sú zapuzdrené a spojené funkčné bloky z knižníc Simulinku. Sieťový prvok bude rozdelený do nasledujúcich blokov:

1. Generátor dát (*vytvorenie dát a určenie ich priority vložením príznakového bytu)*
2. Klasifikácia paketov (*Policer – určenie príslušnosti paketu k triede zaobchádzania*)
3. Aktívne predchádzanie zahlteniu (*zahodenie paketov na základe veľkosti fronty triedy*)
4. Spojovacie pole (*samotné prepojovanie a správa front jednotlivými metódami*)
5. Plánovanie (*určenie poradia paketov, pred odoslaním na HW výstup*)

Jednotlivé bloky tvoria štruktúru, ktorá je zobrazená na obrázku nižšie (Obr. 4.1). Zapojenie realizované v prostredí Matlab/Simulink je uvedené ako príloha A.



Obr. 4.1: *Zapojenie blokov sieťového prvku*

4.2 Generátor dát

Tento blok predstavuje a vytvára pakety viacerých spojení, ktoré by vstupovali do sieťového prvku prostredníctvom vstupných portov. Základným prvkom je blok z knižnice Simulink s názvom generátor entít podľa času (*Time-Based Entity Generator*). Tento blok generuje bloky dát, ktoré môžeme považovať za pakety. Generovanie je možné ovplyvniť nastavením viacerých parametrov, ako je napríklad čas medzi generovaním jednotiek, ktorý môže byť nastavený na konkrétnu hodnotu alebo ovládaný pomocou vstupného portu. Ďalšia voľba dovoľuje voliť typ dátovej jednotky a to medzi prázdnou a štandardnou a taktiež reakciu generátoru na stav blokovania. Prázdna jednotka (*blank*) neobsahuje žiadne atribúty. Štandardná jednotka obsahuje atribúty v poliach s názvom *Priorita* a *Počet.* Tieto polia môžu nadobúdať preddefinovaných hodnôt od 0 do 10.

Pamäť typu FIFO

Na výstup z bloku generátora pripojíme blok (*FIFO Queue*) plniaci úlohu pamäte typu FIFO a veľkosť pamäte nastavíme na desať jednotiek. Veľkosť desať jednotiek nám postačuje na vyrovnanie prípadného zdržania jednotky v nasledujúcich blokoch.

Nastavenie atribútov

Blok, u ktorého môže dôjsť k zdržaniu, je nastavovanie atribútov (*Set Attributes*). Predradenie pamäte je odporúčané v prípade prideľovania viacerých atribútov. Celá dátová jednotka je prijatá a po uložení atribútov je odoslaná na výstup. Požadované atribúty, ich počet, názov a hodnota, sú nastavované pomocou tabuľky v okne nastavenie parametrov bloku. Hodnotu pre atribúty je možné zadávať staticky alebo pomocou hodnoty signálu na pomocnom porte.

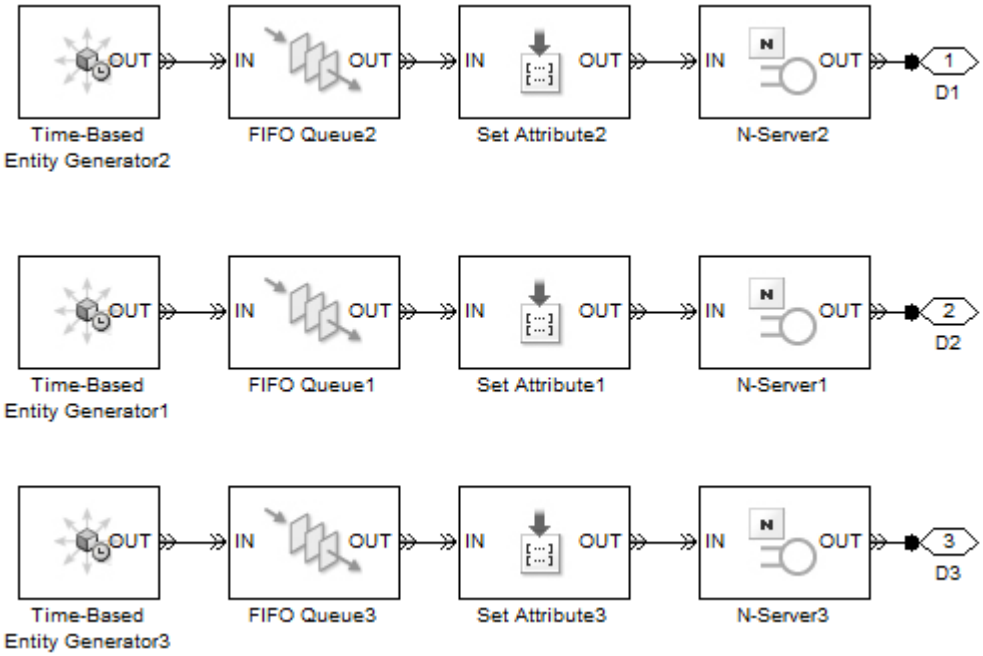
N-Server

Nasleduje blok obsluhy pomenovaný (*N-Server*), ktorý dokáže obslúžiť naraz až *N* entít a s každou jednou pracuje nezávisle určitý čas a potom sa pokúsi ju odoslať cez výstupný port. Doba, po ktorú blok pracuje s dátovou jednotkou je možné pevne nastaviť alebo si túto hodnotu bude niesť v hlavičke ako atribút, prípadne bude určený stavovým portom. V simulácii je použitý spôsob priameho zadania a to na hodnotu 0, pri ktorej je možné sledovať spätné ovplyvnenie generátora situáciou v ďalších blokoch. Pokiaľ je výstupný port blokovaný je naďalej uchovaná v pamäti bloku až kým nie je port uvoľnený.

Združenie dát

Na spojenie vygenerovaných dátových tokov rôznych typov je použitý kombinátor ciest (*Path Combiner*), ktorý v podstate predstavuje multiplex. Počet vstupov zvolíme podľa počtu druhov prevádzky. V prípade doručenia viacerých jednotiek na vstupné porty v rovnakom čase, sa určí poradie podľa súslednosti výstupnej udalosti na predchádzajúcom bloku. Matlab v prípade dvoch udalostí v rovnakom čase simulácie, vykoná tieto úkony postupne, ale ich čas vykonania považuje za rovnaký, aj keď reálny čas je o niečo rozdielny. V prípade, kedy nasleduje fronta a príchod dvoch dátových jednotiek v rovnaký čas by mohol ovplyvniť správanie systému, je ich pozícia vo fronte určená podľa vysielacej sekvencie.

Pravidlo, ktorá dátová jednotka bude mať pri vstupe prednosť a zablokuje výstup pre ostatné, je učený podľa nastavenia precedensu vstupného portu (*Input port precedence*). Ako súčasť generátora by mal kombinovať bloky, bez uprednostňovania, preto použijeme nastavenie cyklického prepínania (*Round-Robin*). Výstupný port môže byť pripojený na pamäť FIFO a N-Server alebo priamo do nasledujúceho bloku navrhnutej logickej štruktúry. Pre potreby kontroly a merania je možné pridať ďalšie štatistické výstupy. V nastavení bloku pod záložkou štatistiky je možné aktivovať dva druhy výstupných portov. Prvým je port označený ako *#d*, na ktorom sa vysielajú údaje o počte odoslaných entít z bloku. Druhým je port označený *last*, ktorý je použitý aj v simulácii a informuje o čísle vstupného portu, z ktorého pochádza vystupujúca entita. Prehľadne zobrazovať tieto údaje je možné pomocou zobrazovacích blokov, konkrétne je použitý (*Signal Scope*) [7].

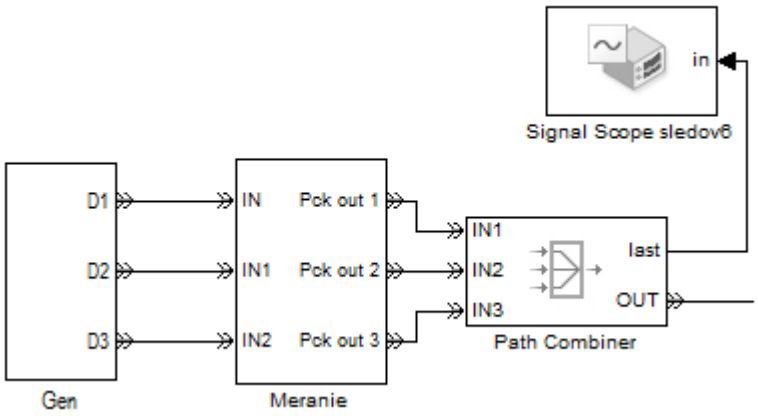


Obr. 4.2: *Zapojenie blokov generátora dát*

<u>Blok merania vyťaženia linky</u>:

Pre zistenie detailnejších informácií o generovaných a prenášaných dátach je nutné použiť zložitejšiu štruktúru. Na kontrolu nám postačia údaje o prenesených entitách (paketoch) za sekundu a množstve bitov prenesených za sekundu. Z dôvodu tejto potreby je zhotovený samostatný blok s názvom Meranie, v ktorom je zapúzdrené samotné zapojenie, pre svoju veľkosť je vložené ako príloha B. Ukážka časti zapojenia, na ktorej je znázornená časť pre meranie celkovej rýchlosti generovania (Paket/s) a jeden blok pre meranie bitovej rýchlosti (bit/s), je predstavená na obrázku 4.4. Celková schéma obsahuje bloky pre meranie bitovej rýchlosti zostávajúcich tried provozu, súčtový člen čiastočných výsledkov a ich zobrazenie. Pre vlastnosti niektorých blokov je nutné zaradenie bloku Meranie pred posledný blok (*Path Combiner*), kompletná štruktúra je znázornená (Obr. 4.3) so zapúzdrením blokov

podieľajúcich sa na tvorbe dát. Na vstupné porty sú pripojené príslušné generované triedy dát a výstupné priamo na kombinátor trasy. Entity sú v bloku merania nezmenené a zdržané len minimálne, prechádzajú iba jedným blokom (*Get attribute*), ktorého výstupný port zároveň predstavuje výstupný port bloku Meranie. Meracie zapojenie spôsobuje oneskorenie zobrazenia výsledkov o jednu nastavenú periódu riadiaceho signálu.



Obr. 4.3: *Celkové zapojenie so zapúzdrením*

Získanie atribútu

Prvým a základným členom v tomto zapojení je blok získania atribútu (*Get Attribute*). Ako napovedá jeho názov, slúži na získanie hodnoty požadovaných atribútov. Nastavenie atribútov, ktoré majú byť získané prebieha rovnako, ako pri bloku nastavovania (Set att.). Podľa poradia v tabuľke sú hodnoty atribútov odosielané na výstupne porty označené $A_1 – A_n$. Použitý bude iba jeden atribút a to veľkosť paketu (*Pcksize*), táto hodnota nám priamo udáva veľkosť v bitoch. Pre zistenie počtu prenesených paketov (entít) použijeme štatistickú vlastnosť bloku, ktorá nám po aktivovaní zobrazí výstupný port *#d*, na ktorý sa odosiela počet odoslaných paketov. Na výstup *OUT* sa odosielajú entity v nezmenenej podobe.

Zbieranie hodnôt

Túto úlohu zabezpečuje blok (*Signal latch*) z knižnice Signál manažment. Je to mnohoúčelový blok pre manipuláciu so signálmi na základe udalosti. V simulácii je použité jednoduché nastavenie, do vnútornej pamäte a zároveň na výstup je zapísaná vstupná hodnota, na základe udalosti. Udalosťou, ktorá vyvolá funkciu zápisu do pamäte je zmena numerickej hodnoty na vstupnom porte *wvc*, ktorý sa zobrazí po zvolení ponuky (*Change in signal from port wvc*) v kategórii nastavení bloku pre zápis do pamäte (*Write to memory upon*). Počiatočná hodnota je nastavená na nulu a reakcia na spôsob zmeny v riadiacom signále na nábeh (*Rising*). Ovládanie čítania z pamäte a odoslanie na výstup je spojené so zápisom a nastavené na hodnotu (*Write to memory event*).

Generátor pulzov

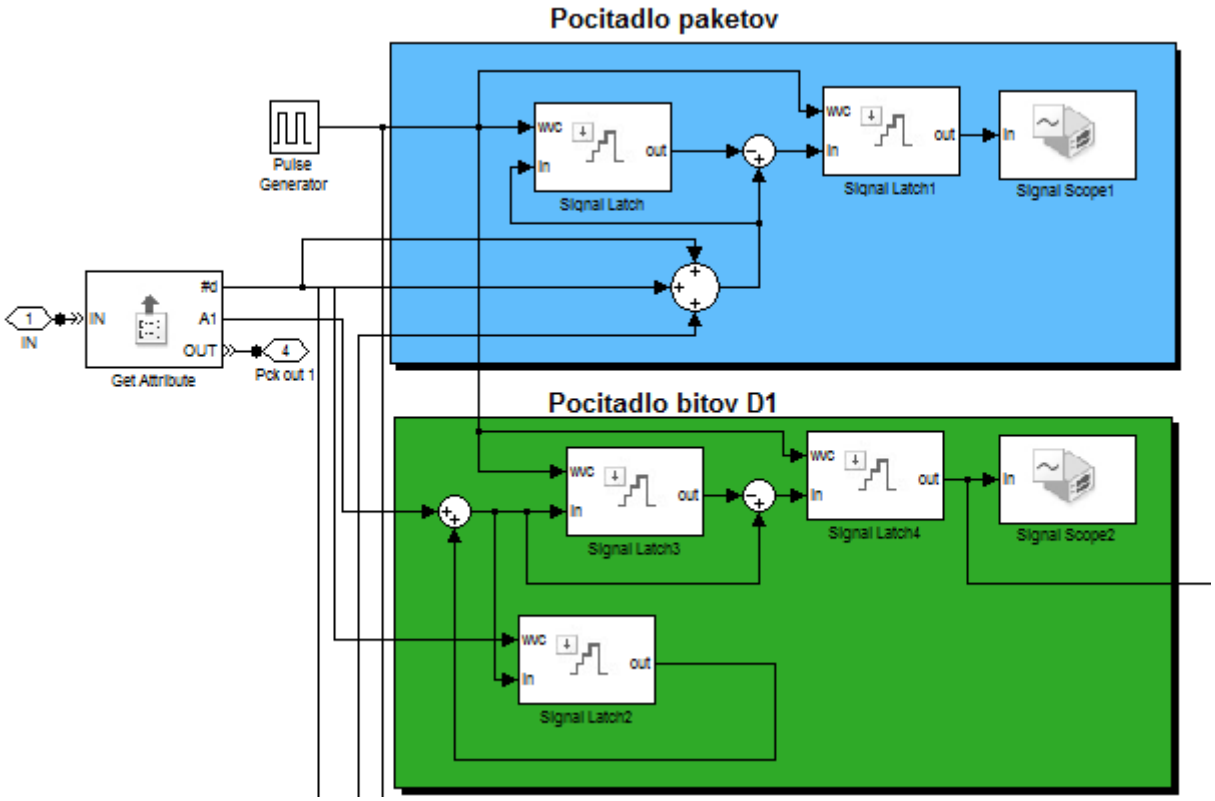
Na riadenie väčšiny blokov prostredníctvom portu *wvc* je použitý jeden generátor obdĺžnikových pulzov (*Pulse Generator*). Najdôležitejšie nastavované hodnoty sú amplitúda, ktorá je už prednastavená na jednotku. Druhým parametrom je čas, ktorý určuje v akých intervaloch bude odčítavaná hodnota, nastavená na jednu sekundu.

Blok sčítania

Viacnásobne použitý v rôznych variantách dvoj a trojvstupových, prípadne ako rozdielový člen.

Zobrazovacia jednotka

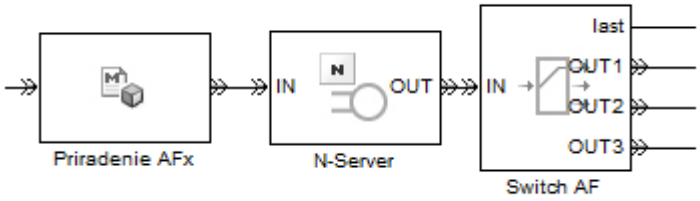
Na zobrazenie výsledných veličín je použitý blok (*Signal Scope*). Vzhľadom na spôsob odčítavania hodnôt je spôsob vykresľovania nastavený na schodovitý. V záložke (*Figure*) sú zadané nastavenia titulov grafov a popisu ôs [7].



Obr. 4.4: *Ukážka zapojenia v bloku Meranie*

4.3 Klasifikácia paketov

Tento blok je zodpovedný za triedenie a označenie prichádzajúcich paketov rôznych služieb do troch vnútorných tried typu zaisteného odoslania (*Assured forwarding*). Pri tomto spôsobe sú prichádzajúce pakety priradené do jednej z troch definovaných AF tried. Tento druh PHB je podrobne popísaný v kapitole 2.4. Priradením do konkrétnej triedy je zohľadnená aj požiadavka na určitú kvalitu služieb QoS. Jednotlivé triedy sa odlišujú vo veľkosti vyrovnávacej pamäte v smerovači, pridelenej šírke pásma výstupného portu a v nastavení výpočtu pravdepodobnosti zahodenia v bloku WRED. Zapojenie blokov popísaných v tejto kapitole je na obrázku 4.5.



Obr. 4.5: *Bloky klasifikácie paketov*

Priradenie AFx

Tento blok (*Attribute function*) je hlavnou častou a vykonáva celú klasifikáciu paketov. Jeho základnou vlastnosťou je pridanie dát do prijatej entity, ktorá je následne odoslaná na výstup. Tieto dáta sú v podobe atribútu, ktorý má svoj názov a hodnotu. Blok dokáže pracovať s existujúcimi atribútmi, prípadne vytvárať nové. Celá funkcia bloku je vyjadrená vloženým matlab kódom, ktorý je písaný v editore otvorenom dvojklikom na tento blok. Použitie kódu v textovej podobe je niekedy oveľa účinnejšie ako grafické vyjadrenie pomocou blokov Simulinku. Tento kód je preložený pomocou externého kompilátoru (*Microsoft visual studio 2008*) do podoby MEX súboru, ktorý predstavuje prepojený podprogram, produkovaný zdrojovým kódom typu C/C++. Tento súbor môže byť spustený v prostredí Matlab rovnako ako vlastné súbory Matlabu a vstavané funkcie. V názve funkcie musia byť atribúty, ku ktorým chceme mať prístup, upravovať ich alebo ich vytvoriť. Zápis kódu použitého v simulácii je uvedení nižšie na obrázku 4.6. Blok pracuje s atribútom *Class*, ktorý označuje typ služby, rozpoznáva osem druhov služieb, ktoré zaraďuje do troch skupín. Nerozpoznané alebo neoznačené pakety sú zaradené do najnižšej triedy zaobchádzania. Vytvorený a pridaný je nový atribút *AF*, ktorý je používaný vo vnútornej štruktúre smerovača.

N-Server

Blok a jeho funkcia je popísaná v predchádzajúcej podkapitole. Rozdiel v nastavení je v položke *Number of servers*, ktorá je nastavená na číslo päť, čo poskytuje rezervu pre prípadnú potrebu obslúžiť viacej prichádzajúcich dátových jednotiek v jednom okamihu.

Switch AF

Funkcia tohto bloku (*Output Switch*) je prepínanie entít z jedného vstupného portu na výstupné porty, podľa určitého kritéria. V simulácii je blok použitý na rozdelenie zatiaľ spojeného toku všetkých paktov patriacich do troch AF tried. Z tohto dôvodu je počet výstupných portov (*Number of entity output ports*) nastavený na tri a výber výstupného portu (*Switching criterion*) je na základe atribútu *AF* v pakete. Pokiaľ nie je vybraný port blokovaný, paket je odoslaný. V záložke štatistík je možné zapnúť ďalšie porty, pre kontrolu a vyhodnotenie. Použitý je iba port s označením *last*, na ktorý sú posielané údaje o naposledy použitom porte. Následne už rozdelené toky paketov vstupujú do časti AQM, kde je implementovaná technológia WRED.

```
function [out_Class, out_AF] = fcn(Class)
%FCN    Access and modify attributes with Embedded MATLAB Function
%
% Input arguments must match the attribute names.
% Output arguments must be out_<AttributeName>,
% e.g., out_ServiceTime where ServiceTime is the attribute name.
%
% The output attribute will be created if it is not present.
%
switch Class
    case 1
        out_AF = 1;
    case 2
        out_AF = 1;
    case 3
        out_AF = 2;
    case 4
        out_AF = 2;
    case 5
        out_AF = 2;
    case 6
        out_AF = 3;
    case 7
        out_AF = 3;
    case 8
        out_AF = 3;
    otherwise
        out_AF = 3;
end
out_Class = Class;
```

Obr. 4.6: *Kód z bloku Attribute function*

4.4 WRED

Nasledujúca časť poskytuje aktívny manažment front AQM a konkrétne technológiu WRED. Táto technológia bola popísaná v priebehu prvej kapitoly, konkrétne v časti 1.4 Riadiaci člen. Zahadzovanie paketov, na základe nastaveného profilu pre jednotlivé AF triedy, ako aj zahadzovanie pri plnej fronte (*Tail drop*), je realizované prepínačom výstupných portov. Konkrétny mechanizmus prepočtu pravdepodobnosti zahodenia pre danú triedu je zapúzdrený v podsystéme *WRED riadenie*. V tejto časti sú všetky ostatné podsystémy tvorené jedným druhom bloku a preto budú popísané priamo, ich príslušnosť k podsystému bude uvedená v zátvorke. Výnimkou je podsystém *WRED prepočet priemeru fronty*.

Plánovanie TO (*Nastavenie TO*)

Využité sú kusy bloku (*Schedule Timeout*) na nastavenie času, ktorý môže paket stráviť v prepínači. Toto opatrenie poskytuje metódu zahadzovania plnej fronty (*Tail drop*). Nastavený je identifikátor (*Timeout tag*) na hodnotu TO1, číslo odpovedá triede AF. Zmena oproti prednastaveným hodnotám je ešte v poli intervalu čakania (*Timeout interval*), kde je nastavená hodnota 10ms. Táto hodnota je dostatočne malá, aby sa to mohlo považovať za okamžité zahodenie v prípade plnej fronty a zároveň poskytuje dostatočný čas pre príchod ovládacieho signálu na port *p*, z podsystému *WRED riadenie*.

Odstránenie TO (*Rusenie TO*)

Rovnako sú použité tri bloky (*Cancel Timeout*) na odstránenie udalosti vypršania času (*timeout*). V každom bloku je nastavené príslušné označenie (*Timeout tag*), ktoré má byť v prechádzajúcich paketoch odstránené.

Zahodenie (*DROP*)

Tento blok (*Entity Sink*) predstavuje cieľ pre entity a dochádza k ich zničeniu. Ide o preventívne opatrenie, aby nedošlo k hazardným stavom v programe Simulink.

Switch drop AFx

Rovnako sa jedná o prepínač výstupných portov (Output Switch), ale s rozdielnym nastavením. Počet výstupných portov (*Number of entity output ports*) je nastavený na dva a výber výstupného portu (*Switching criterion*) je riadený pomocou signálového portu *p*. Následne je zaškrtnutá voľba na uloženie paketu pred prepnutím (*Store entity before switching*). V druhej záložke (*Timeout*) je zaškrtnutá ponuka na zobrazenie portu pre pakety, ktorým vyprší čas uloženia (*Enable TO port for timed-out entities*). Cez tento port budú odchádzať pakety určené na zahodenie z dôvodu plnej fronty. V poslednej záložke (*Statistics*) je zapnutý port, ktorý informuje o počte odoslaných paketov *#d*. Tento údaj je použitý na výpočet v podsystéme *WRED riadenie.*

Blok WRED prepočet priemeru fronty

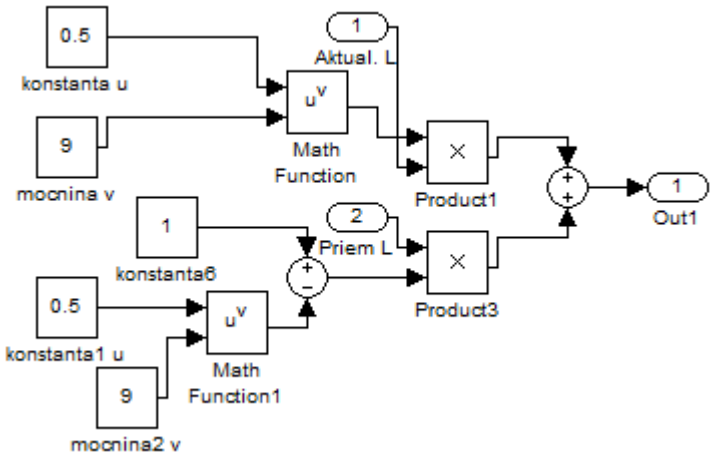
Tvorí prechodový blok, v ktorom je prevedený výpočet priemeru zaplnenia fronty pre každú triedu AF. Podľa špecifikácie technológie WRED nepracuje priamo s priemernou dĺžkou fronty, ktorá by spôsobila určité oneskorenie reakcie. Priemerná hodnota logicky stúpa aj klesá pomalšie ako skutočná hodnota a preto bol vytvorený spôsob prepočtu, ktorý zahrňuje aj aktuálnu hodnotu naplnenia fronty. Priemer je počítaný podľa nasledujúceho vzorca:

$$priemer = \left(priemerné\_naplnenie * \left(1 - \left(\frac{1}{2}\right)^{n}\right)\right) + \left(aktuálna\_veľkosť * \left(\frac{1}{2}\right)^{n}\right) \qquad (4.1)$$

Premenná *n* v tomto vzorci môže byť menená užívateľom. Predstavuje exponenciálny vážený faktor, ktorý vyjadruje vplyv priemerného naplnenia fronty na výsledok. Pri príliš nízkej hodnote môže dochádzať k zbytočnému zahadzovaniu paketov a pri príliš vysokej hodnote nemusí WRED reagovať na zahltenie. Základné nastavenie *n* je na hodnotu deväť, ktorá je použitá aj v simulácii. Jediným zložitejším blokom v zapojení výpočtu je blok *Math* a jeho nastavenie, ostatné bloky a ich nastavenie je čitateľné z obrázku 4.7.

Math Function

Je použitý na výpočet mocniny zo vzorca. Pre toto použitie je potrebné vybrať z ponuky funkcií (*Function*) ponuku *pow*, ktorá prevádza umocnenie jedného vstupu, hodnotou druhého vstupu. Prvý argument *u* je horný vstup alebo pri pootočení ľavý vstup, následne zostávajúcim je argument *v*. Zvyšné nastavanie zostávajú nezmenené.



Obr. 4.7: *Prepočet priemeru pre WRED*

Blok WRED riadenie

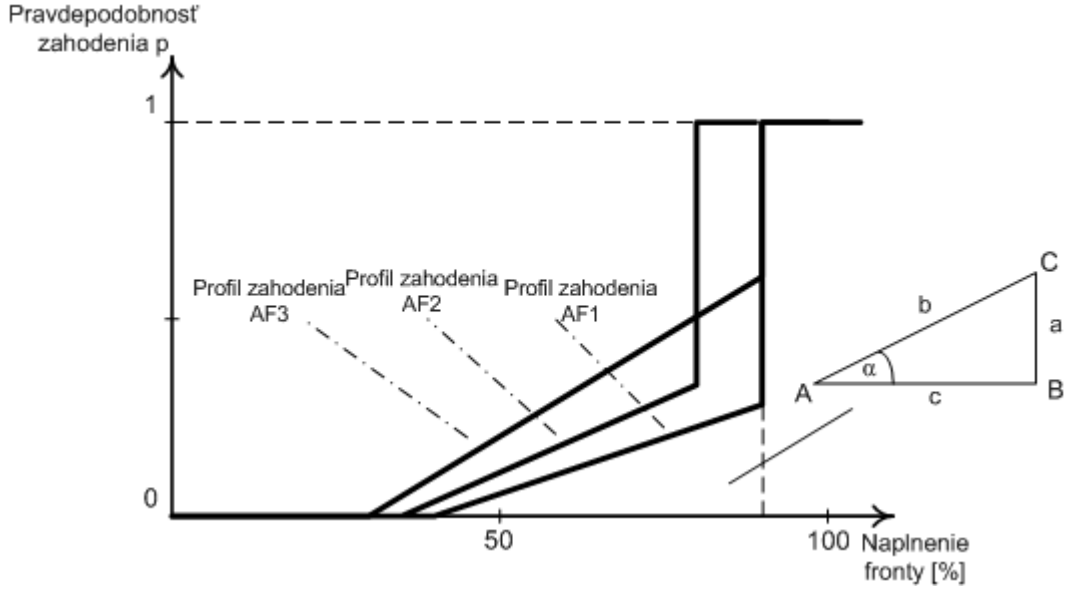
Vykonávanie výpočtu pravdepodobnosti zahodenia, a teda vysielanie impulzu na signálový port *p* prepínača výstupov a samotné nastavenie profilu zahadzovania, je realizované zapojením v tomto bloku. Do bloku vstupujú dva druhy parametrov, a to priemerná dĺžka fronty z *FIFO Queue AFx* a počet odoslaných paketov cez prepínač *Switch drop AFx.* Následne sú vložené tri zapojenia, každé pre jednu triedu AF.

Prvým krokom je podelenie priemernej dĺžky fronty konštantou, ktorá odpovedá maximálnej dĺžke fronty. Následne je získane percentuálne naplnenie fronty vynásobením koeficientu číslom sto. Výsledná hodnota je privedená na vstupy troch blokov, bloky porovnávajúce vstup s definovanou konštantou predstavujú spodný prah α_{min} a horný prah β, kde pri splnení podmienky je na výstupe bloku hodnota jedna. Tretím blokom je rozdielový člen, kde je od percentuálnej hodnoty naplnenia fronty odpočítaná hodnota spodného prahu. Výsledok je podelený číslom sto, čím dosiahneme zmenu mierky od 0 do 1 odpovedajúcu *y* ose pravdepodobnosti zahodenia. Hodnota je ešte násobená prevodovou konštantou Θ vyjadrujúcou pomer medzi stranami *a* a *c* pravouhlého trojuholníka definovaného profilom zahodenia (Obr. 4.8). Výsledný pomer medzi stranami je vypočítaný pomocou sínusovej vety.

$$\frac{a}{\sin\alpha} = \frac{c}{\sin\gamma} \Rightarrow a = c \cdot \frac{\sin\alpha}{\sin\gamma} \tag{4.2}$$

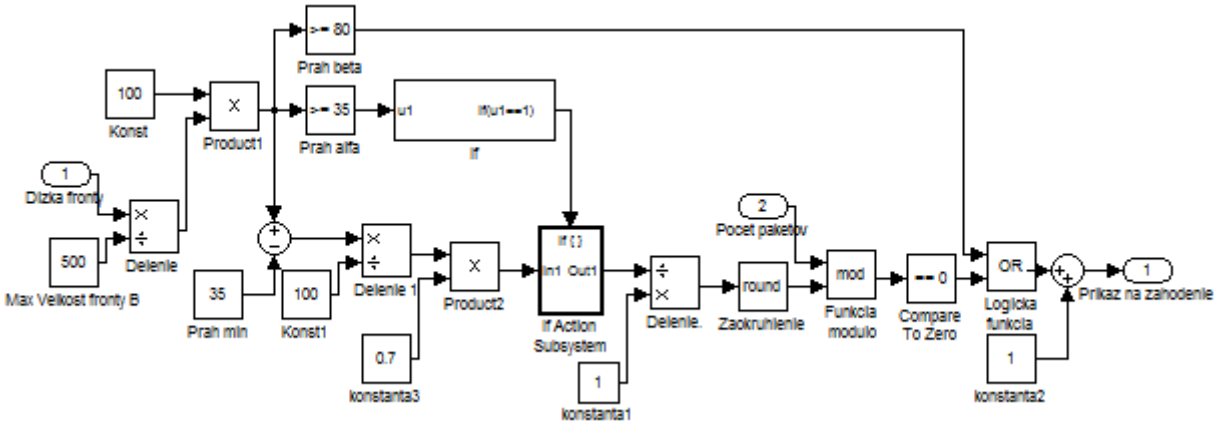
Profily sú pre jednotlivé triedy AF nastavené nasledovne:

- **AF1**: spodný prah α_{min} = 40 [%], β = 90 [%] a uhol zdvihu α = 30 [°]. Vypočítaný koeficient Θ = 0,58
- **AF2**: spodný prah α_{min} = 35 [%], β = 80 [%] a uhol zdvihu α = 35 [°]. Vypočítaný koeficient Θ = 0,7
- **AF3**: spodný prah α_{min} = 30 [%], β = 90 [%] a uhol zdvihu α = 45 [°]. Vypočítaný koeficient Θ = 1



Obr. 4.8: *Profily zahodenia tried AF*

Hodnota je prenesená cez akčný podsystém funkcie If, ktorý slúži ako brána, ktorá sa otvorí len pri splnení podmienky dosiahnutia minimálneho prahu α_{min}. Vstupuje do bloku delenia a jej hodnota je obrátená 1/vstup a zaokrúhlená v nasledujúcom bloku na najbližšie celé číslo. Táto hodnota predstavuje poradové číslo paketu *n*, ktorý má byť zahodený. Medzi počtom odoslaných paketov, ktorých hodnota je privedená z vstupného portu dva a výsledkom z prechádzajúceho bloku je prevedená operácia modulo. V prípade zhody alebo delenia bez zvyšku (celočíselný násobok *n*) je na výstupe nula, ktorá je porovnávacím blokom prevedená na jednotku, ktorá vstupuje do logického členu OR spolu s hodnotou vypovedajúcou o dosiahnutí vrchného limitu β. V sčítacom bloku je jednotka vyjadrujúca povel na zahodenie spočítaná s konštantou jedna a tým vznikne signál hodnoty dva, čo prepne port prepínača výstupov *Switch drop AFx* a paket je zahodený. V zapojení sú použité nasledujúce bloky z knižníc Simulinku.



Obr. 4.9: *Zapojenie profilu WRED pre AF2*

Konštanta

Poskytuje na svojom výstupe konštantnú hodnotu odpovedajúcu nastavenej hodnote v poli (*Constant value*), v zapojení je použitých viac kusov toho bloku s individuálnym nastavením hodnoty. Nastavenie hodnôt je možné odsledovať na zobrazení na obrázku 4.9. Iné parametre neboli menené oproti prednastaveniu bloku.

Násobenie, delenie

Základom týchto blokov je jeden blok (*Product*), ktorý je modifikovaný zápisom do riadku (*Number of inputs*) symbolu / pre vstup deliteľa a symbolu * pre vstup delenca, v prípade jednoduchého násobenia sa udáva iba počet vstupov číselnou hodnotou.

Porovnanie s konštantou

Porovnáva vstupnú hodnotu s vnútorne definovanou konštantou a v prípade splnenia podmienky vysiela na výstup hodnotu jedna. Výber je možný z pomedzi štandardných matematických operátorov. V poli (*Constant value*) je možné zadať hodnotu, voči ktorej bude posudzovaný vstup. Nastavenie konštánt je dobre viditeľné na obrázku zapojenia.

Funkcia IF

Skladá sa z dvoch blokov, a to z bloku IF (*If*) a z bloku prevádzajúceho akciu (*If Action subsystem*). V kombinácii predstavujú ekvivalent štandardný v logike jazyka C. IF blok je zodpovedný za podmienky a kontrolu vstupu. Podsystém akcie vykonáva na príkaz nadradeného bloku priamo jednoduchú funkciu. Blok podmienok (*If*) umožňuje nastavovať počet vstupných portov a celú sadu podmienok (*if, elseif, else*), každá táto podmienka sa premietne ako samostatný ovládací port. V riadení WRED je použitá jednoduchá funkcia, kedy v prípade, že sa vstup rovná jednej (*if u1==1*) je povolené vykonanie funkcie podsystému. V bloku je od značená ponuka na zobrazenie druhého portu (*else*). Funkcia podsystému je jednoduchá brána, ktorá po príchode riadiaceho impulzu povolí prechod signálu. Pre naše potreby je zmenené nastavenie výstupného portu z udržania hodnoty aj po vypnutí (*held*) na nulovanie výstupu (*reset*). Počiatočná hodnota výstupného portu je zmenená na nulu v poli (*Initial output*).V Podrobný popis všetkých možností a iných nastavení je uvedení v literatúre [7].

Zaokrúhlenie

Zaokrúhlenie vstupnej hodnoty a jej následné odoslanie na výstup. Blok (*Rounding*) poskytuje viacej možností zaokrúhlenia:

- Floor – vstupný signál je zaokrúhlený na najbližšie celé číslo smerom dole.
- Ceil – vstup je zaokrúhlený na najbližšie celé číslo smerom hore. (+∞)
- Round – zaokrúhlenie na najbližšie celé číslo, bez rozdielu smeru
- Fix – zaokrúhlenie na najbližšie celé číslo smerom k nule

Je použité nastavenie *round*, ktoré najlepšie zodpovedá spôsobu použitia v zapojení.

Funkcia modulo

Poskytuje možnosť využiť viacero zložitejších matematických funkcií. V bloku (*Math*) je zvolená v položke (*Function*) funkcia modulo. Typ výstupného signálu (*Output signal type*) je nastavený na reálne číslo (*real*). Zvyšok nastavení je nezmenených.

Logická funkcia

Základný blok (*Logical Operator*) je obdoba predchádzajúceho bloku, s rozdielom, že poskytuje na výber všetky logické funkcie namiesto matematických. Zvolená je funkcia (*Operator*) OR, teda výstup je jednotka v prípade, že aspoň jeden zo vstupov je v jednotke.[7]

4.5 Fronty, smerovanie a plánovanie

Táto časť je tvorená prevažne podsystémami, samostatne vystupujú iba fronty, ktoré sú tvorené pamäťou typu FIFO. Každá trieda AF je ukladaná do vlastnej fronty, ktoré sú prispôsobené svojou kapacitou. Pakety po výstupe z fronty vstupujú do spojovacieho poľa (*Spojovacie pole Smerovanie*), v ktorom prebehne určenie výstupného portu podľa adresy v pakete. Na jednom porte je definovaných viacero dosiahnuteľných adries. Plánovač (*Scheduler*) pracujúci nad konkrétnym portom registruje prítomnosť paketu a povolí odoslanie. V prípade viac ako jedného paketu s požiadavkou na odoslanie na daný port, sa prevedie určenie pomeru veľkosti paketu k vyhradenej šírke pásma pre danú triedu AF a následne sa určí poradie odosielania paketov. Druhý paket čaká v pamäti, pokiaľ nie je ukončené odosielanie prvého a tretí na prvý a druhý. O získanie hodnoty veľkosti paketu a povolenie na vysielanie sa stará druhá časť spojovacieho poľa (*Spojovacie pole Odosielanie*). Každý paket je odosielaný plnou rýchlosťou, ktorú dosahuje linka na danom porte.

FIFO Queue AFx

Samostatne stojaci blok (*FIFO Queue*) z ponuky knižníc Simulinku. Jednotlivé bloky sa líšia v nastavení veľkosti fronty (*Capacity*). Pre triedu AF1 zodpovedá veľkosti tisíc paketov, pre triedu AF2 päťsto paketov a pre triedu AF3 je nastavená hodnota na sto paketov. Spoločné nastavenie predstavuje zapnutie portu pre údaj priemernej dĺžky fronty (*Average queue length, len*), s týmto údajom pracuje mechanizmus WRED.

Spojovacie pole – Smerovanie

Paket vstupuje do podsystému, kde je z neho v prvom bloku (*Get Attribute*) získaná cieľová adresa, ktorá je odoslaná sa ďalšie spracovanie. Ďalším spracovaním sa myslí porovnanie s tabuľkou adries a určenie portu, na ktorý sa má odoslať pomocou zápisu parametru *Port* do paketu. Zapísanie parametru je prevedené nasledujúcim blokom (*Set Attribute*). Následne je tento paket prepnutý na príslušný port v prepínači výstupných portov (*Output Switch*), podľa hodnoty parametru *Port*. Na každý výstupný port je ešte pripojená pamäť FIFO, ktorá má veľkosť iba jeden paket a slúži na detekciu čakajúcich paketov pre Plánovač. Zapojenie je znázornené na obrázku 4.10, každá trieda AF je smerovaná jedným vyhotovením tohto zapojenia.

Get Attribute2

Blok slúži na získanie cieľovej adresy z paketu a odoslaniu tohto údaju. Zisťuje iba parameter *Adresa*, inak je v základnom nastavení. Tento blok a manipulácia s ním je podrobne rozobratá na začiatku kapitoly (Získanie atribútu).

Set Attribute1

Nastavuje hodnotu do atribútu *Port*, ktorý rovnako vytvára v záhlaví paketu. Nastavovaná hodnota sa dynamicky mení podľa signálu na vstupnom porte A1. Toto je zaistené nastavením *Signal port* v stĺpci (*Value From*). Podrobnejší popis je podobne ako pre Get Attribute uvedený na začiatku kapitoly.

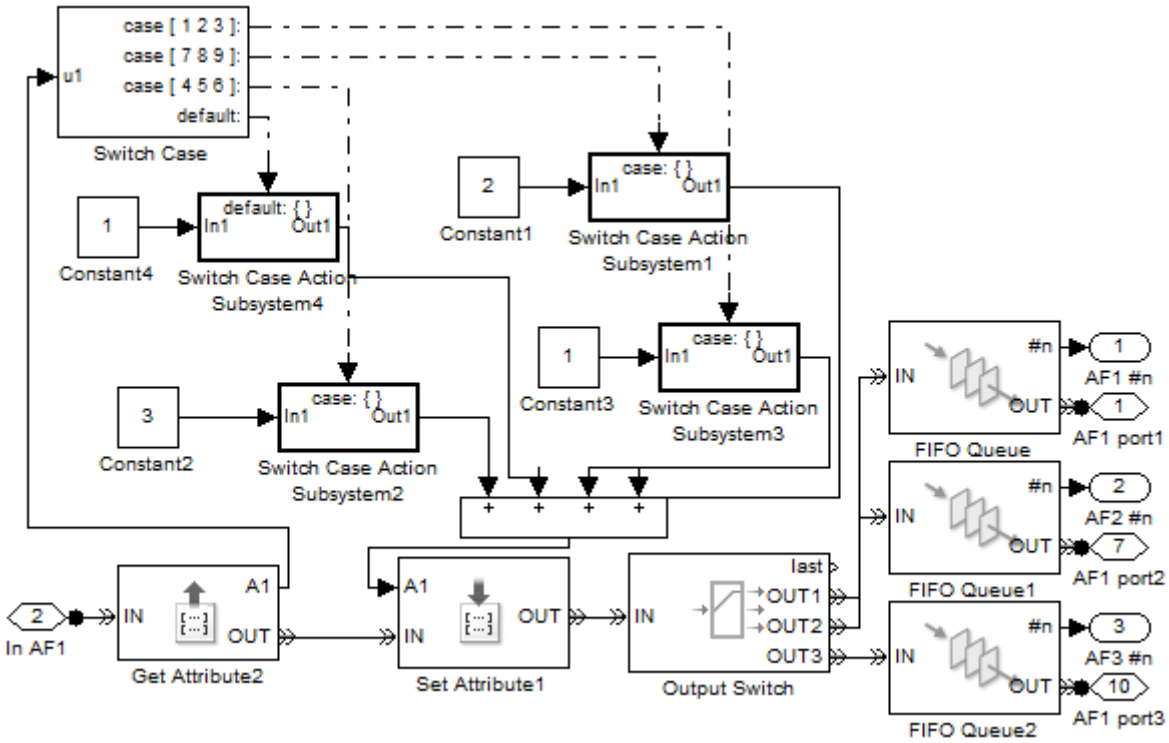
Output Switch

Prepína medzi tromi výstupmi na základe atribútu *Port*. Nastavenie sa prevedie zvolením ponuky *From attribute* medzi ponukami (*Switching criterion*) a zadaním názvu atribútu do príslušného riadku (*Attribute name*).

Switch Case

Rovnako ako ostatné funkcie prevzaté z programovacích jazykov, je aj táto tvorená dvomi blokmi. Ovládací uplatňuje podmienku Switch na vstupnú hodnotu a následne aktivuje blok vykonávajúci akciu (*Switch Case Action Subsystem*). Podmienky pre jednotlivé prípady (*case*) sa definujú v jednom riadku (*Case conditions*). Táto skutočnosť vyžaduje rozlíšenie jednotlivých prípadov pomocou hranatých zátvoriek. Využitý je bunkový systém a preto sú všetky podmienky zahrnuté do množinových zátvoriek, celkový zápis nastavenia vyzerá nasledovne *{[1,2,3],[7,8,9],[4,5,6]}*. V tomto zápise predstavujú čísla konkrétnu adresu a ich združenie v hranatých zátvorkách predstavuje port. V prípade, že nie je nájdená zhoda, je aktivovaný štvrtý výstup (*default*), ktorý je možné považovať za defaultnú bránu.

V blokoch akcie je nastavená počiatočná hodnota na *0* a hodnota výstupu v prípade neaktívnosti (*Output when disabled*) je nastavená na *reset*. V momente príchodu riadiaceho impulzu je prevedená vstupná hodnota konštanty na výstup.



Obr. 4.10: *Zapojenie smerovacej funkcie*

FIFO Queue

Fronta slúži len na detekciu paketu a jej veľkosť je nastavená na jeden paket. V poslednej záložke (*Statistics*), je na funkciu detekcie povolené vysielanie počtu paketov vo fronte (*Number of entities in queue*). Tento signálový port je označený simbolom *#n*.

Spojovacie pole – Odosielanie

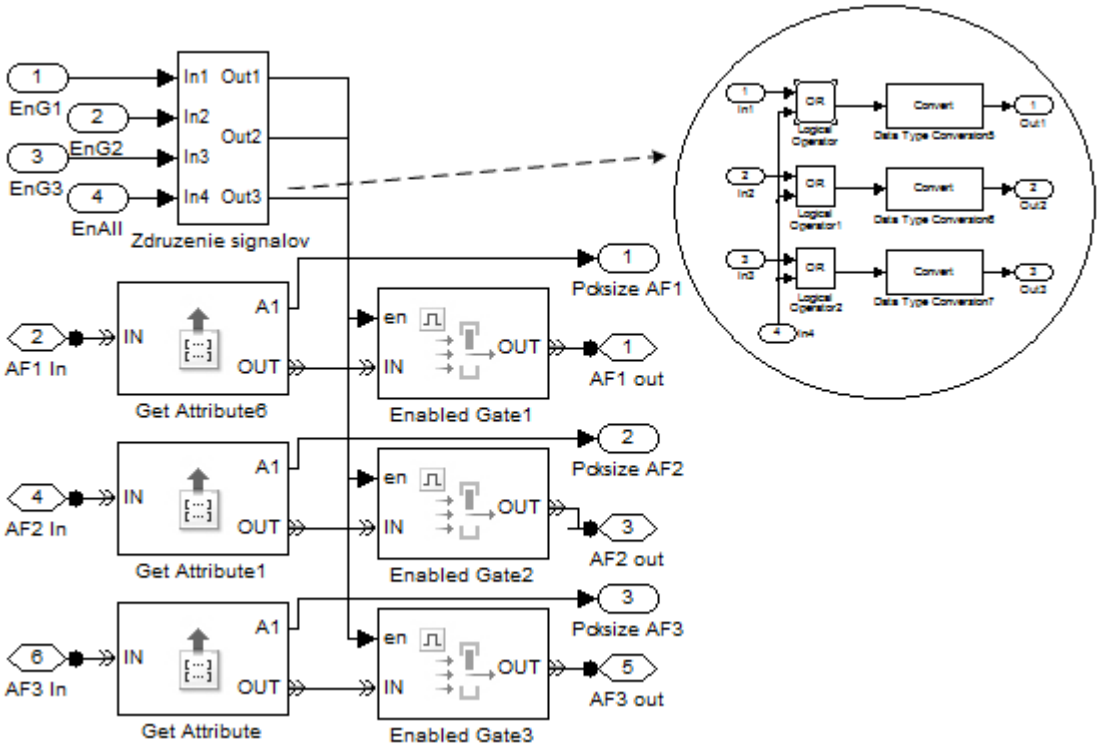
Hlavnou úlohou je povolenie odoslania paketu v správnu dobu. Získava pre plánovač údaje o veľkosti paketov. Po vstupe paketov sú z nich získané údaje a čakajú na otvorenie odchodzej cesty od plánovača. Obsahuje vnorený podsystém *Združenie signálov*, ktorého jediná funkcia je spojiť ovládacie signály z plánovača pre jednotlivé triedy so signálom povolenia vysielania v prípade čakania iba jedného paketu.

Get Attribute

Blok z knižnice Simulinku, ktorý už bol popísaný v predchádzajúcich častiach. Líšia sa parametrom, ktorého hodnotu získavajú a to je konkrétne *PcksizeX*. Veľké *X* predstavuje číslo triedy AF, a teda je z množiny od jedna po tri.

Enable Gate

Je to jednoduchá brána, ktorá povolí alebo zamietne príchod paketu na základe povolovacieho signálu na ovládacom porte *en*. Vstupný port je dostupný pre paket v prípade pozitívneho signálu na ovládacom porte. Prichádzajúci paket je okamžite preposlaný na výstup.



Obr. 4.11: *Spojovacie pole - Odosielan*

Podsystém Združenie signálov

V podsystéme sa nachádzajú iba už popísané logické funkcie OR a blok slúžiaci na konverziu dátového typu (*Data Type Conversion*) v rámci simulinku, kde môže nastať problém s kompatibilitou. V týchto blokoch nie sú k dispozícii takmer žiadne nastavenia, a preto je ich ďalší popis bezpredmetný. Vnútorné zapojenie je zobrazené spolu s celkovým zapojením na obrázku 4.11.

Scheduler

Tento podsystém rieši problém odosielania a plánovania paketov. V prípade iba jedného paketu iba jednej triedy AF na odoslanie, nedôjde k výpočtu, ktorý by bol zbytočný a rovno je vydaný pokyn na odoslanie. V prípade čakania paketov dvoch alebo troch tried, prebehne určovanie poradia. Na určenie počtu čakajúcich paketov slúži údaj z pamäti v bloku *Spojovacie pole – Smerovanie*. Výsledné poradie je určené na základe veľkosti paketov, ktoré poskytuje blok *Smerovacie pole – Odosielanie* sú podelené šírkou pásma pridelenou jednotlivým triedam AF. Získané indexy sú usporiadané od najmenšieho po najväčší a odoslané do príslušných vetiev zapojenia. Najnižší index podstúpi vo svojej vetve iba určovanie, o paket ktorej triedy sa jedná a hneď je vyslaný príkaz na odoslanie. Druhý index je rovnako podrobený testovaniu a čaká na signál o odoslaní prvého, po ktorom je zahájené jeho odoslanie. Tretí index po zatriedení čaká na dva impulzy a až potom je povolené jeho odoslanie. Na predídenie nesprávnej identifikácie impulzov musia byť signály privedené selektívne a to systémom: signál o odoslaní AF1 a AF2 na bloky rozhodujúce o odoslaní paketu AF3, pre prípad druhého a tretieho v poradí. Rovnakým spôsobom aj pre ostatné triedy AF. Schéma zapojenia je pre svoju rozľahlosť a v záujme prehľadnosti uvedená ako príloha C.

AF 1,2

Je to blok vstupného prepínača (*Switch*) s riadiacim portom. Pripojený je port tri, až po splnení podmienky na porte dva je pripojený port jedna. Hodnota na rozhodovacom porte číslo dva je porovnávaná s prednastavenou prahovou hodnotou (*Treshold*). Táto hodnota je nastavaná na jednotku a podmienka (*Criteria for passing first input*) v podobe, hodnota na vstupe dva musí byť väčšia alebo rovná prahovej hodnote. Prepínač slúži na privedenie impulzov o odoslaní ich presného smerovania do bloku, ktorý na základe nich rozhoduje.

Aktivácia plánovača

Jedná sa o blok (*If*), ktorý v prípade, že je čakajúci iba jeden paket obíde plánovač a povolí odoslanie všetkých tried AF. Vo frontách je iba paket jednej triedy AF a teda nedôjde ku kolízii. V prípade viac ako jedného čakajúceho sa použije plánovač na určenie poradia. Popis tohto bloku už bol uvedený, a preto nasledujú iba nastavenia podmienky (*If expression*). Podmienka je *u1 > 1*, ak vstup presiahne hodnotu jedna, vydá pokyn akčnému podsystému, v inom prípade je aktivovaný výstup *else*.

Multiplex

Slúži na spojenie výsledkov delenia veľkosti jednotlivých paketov do jedného vektora, ktorý vstupuje do bloku riadenia. Tento obvod (*Mux*) neobsahuje žiadne nastavenia okrem počtu vstupov.

Triedenie

Pracuje s vektorom vytvoreným multiplexom a usporadúva prvky v ňom zvolenom móde. Blok (*Sort*) je nastavený na zoradenie prvkov vzostupne (*Ascending*), v položke druh triedenia (*Sort order*). Následne sa využíva index prvku a nie jeho hodnota, ktorá je pre určenie triedy AF nepodstatná. Preto je použité nastavenie v položke mód (*Mode*) na výstup typu hodnota a index (*Value and index*). Výstup hodnota je nepripojený a slúži v pre prípad záujmu na pripojenie zobrazovacej jednotky.

Demultiplex

Tento blok (*Demux*) má opačnú funkciu ako multiplex a rozdeľuje jednotlivé prvky vo vektore, keďže je pripojený na výstup indexov rozdelí indexy prvého, druhého a tretieho paketu, ktoré sú privedené do príslušných vetiev.

Switch case

Slúži na určenie, o ktorú triedu AF paketu sa jedná a podľa toho je aktivovaný podsystém pre príslušnú bránu. Podľa poradia indexu sú obvody akčného podsystému zložitejšie. V prípade prvého indexu je priamo odoslaný signál na povolenie odoslania, pri druhom je odoslanie podmienené jedným impulzom o odoslaní a pre tretí je v obvode zaradené počítadlo, ktoré povolí odoslanie až po dvoch signáloch o odoslaní. Nastavenie a podrobnejší popis sú zhodné s už uvedenými v časti Spojovacie pole - Smerovanie.

Počítadlo

Je tvorené jedným blokom (*Counter*), ktorý je nastavením upravený tak, aby zvýšil svoju hodnotu o jedna, smerom hore (*Count direction - up*) a reagoval (*Count event*) iba na nenulové pulzy (*Non-zero sample*). Maximálna hodnota (*Maximum count*) je nastavená na číslo jedna a počiatočná hodnota (*Initial count*) na nulu. S príchodom prvého impulzu sa nastaví na číslo nula, s príchodom druhého na jednotku a po treťom znova na nulu. Teda s príchodom každého druhého pulzu je povolené odoslanie. Vždy s príchodom druhého oznámenia odoslania je odoslaný aj tretí paket.

Ďalšie použité bloky

Zostávajúce bloky, ktoré boli použité už boli popísané a ich jednoduché nastavenie je zrejmé aj zo zapojenia. Sú to bloky: Konštanta, Porovnanie s konštantou, konvertor dátového typu a logické funkcie OR a AND. [7]

4.6 Fyzické rozhranie liniek

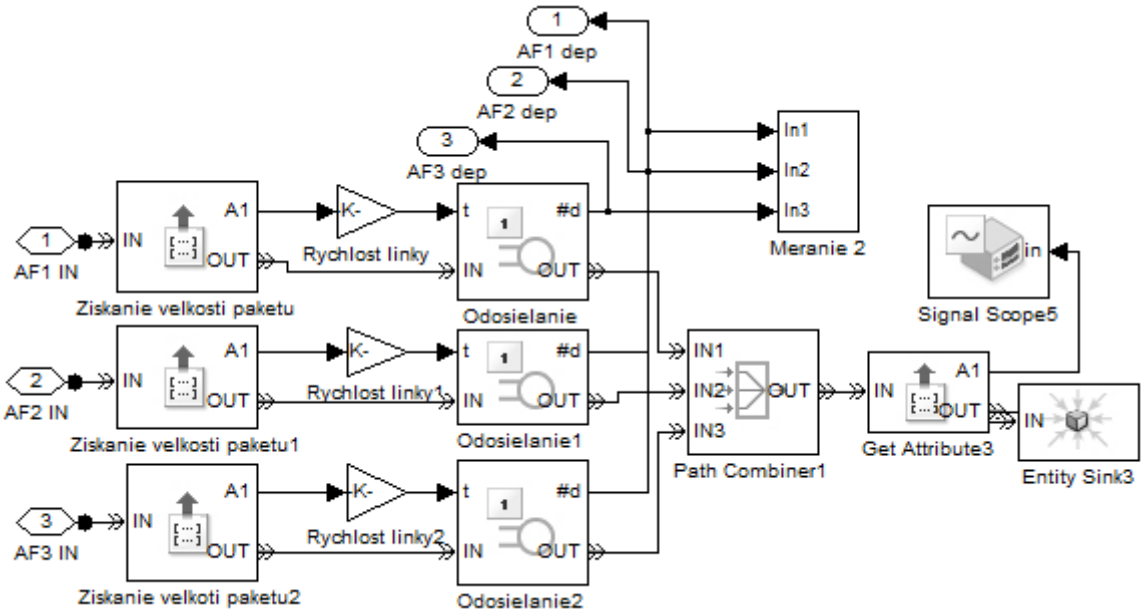
Linku predstavuje jednoduché zapojenie, kde je definovaná maximálna rýchlosť linky a je počítaný čas, na ktorý bude využitá odosielaním daného paketu. Po dokončení odosielania paketu je vyslaný informačný signál do bloku plánovača a začína vysielanie nasledujúceho paketu. Podsystém *Meranie 2* slúži na zistenie paketovej rýchlosti jednotlivých tried AF na výstupnej linke. Zapojenie je rovnaké ako časť *Počítadlo paketov* uvedené v podkapitole Generovanie dát.

Získanie veľkosti paketu

Jedná sa o blok získania atribútu (*Get attribute*) a zisťuje sa veľkosť aktuálne prichádzajúceho paketu *PcksizeX*. Písmeno *X* predstavuje číslo triedy AF. Paket aj hodnota atribútu je poslaná do nasledujúcich blokov. Všeobecný popis bloku je uvedený v predchádzajúcich podkapitolách.

Rýchlosť linky

Blok (*Gain*) poskytuje možnosť definovať rýchlosť linky, ktorou je rovno podelená veľkosť paketu a na výstupe dostaneme rovno čas, ktorý potrebuje paket na svoje odoslanie. Rýchlosť linky sa definuje do poľa (*Gain*) v tvare *1/(rýchlosť linky * n),* výsledok na výstupe je násobok veľkosti paketu a takto zadanej rýchlosti linky. Do vzorca vstupuje ešte koeficient *n,* vyjadrujúci časť šírky pásma pridelenej príslušnej triede AF. Z výberu druhov násobenia (*Multiplication*) je zvolená možnosť *Element-wise(K.*u).*



Obr. 4.12: *Fyzické rozhranie linky*

Odosielanie

Blok (*Serialization Delay*) prijme paket a pozdrží ho v časovom úseku, ktorý je definovaný vstupným portom *t*. Na tento časový port prichádza údaj z bloku Rýchlosť linky. Pokiaľ je výstupný port blokovaný, paket je uchovaný v bloku, táto možno ale v zapojení nemôže nastať. Pre zobrazenie vstupného časového portu *t* je potrebné vybrať z ponuky

(*Service time from*) možnosť *Signal port t*. Pre signalizáciu plánovaču a meranie je potrebné povoliť štatistický port udávajúci počet odoslaných paketov (*Number of entity departed*).

Path Combiner

Slúži na spojenie jednotlivých dátových tokov a na jeho výstupe je už jedna linka aj fyzicky, nie len v logickej forme. Nasledujúce bloky slúžia už len pre testovanie a zobrazenie výsledkov. Predstavujú priamo zakončenie linky na jej druhom konci.

4.7 Nastavenie parametrov a spustenie simulácie

Pred spustením samotnej simulácie je potrebné zadať do *Workspace* Matlabu hodnoty, s ktorými bude pracovať Generátor dát. Vyžadované hodnoty sú pre urýchlenie práce definované v súbore *Generátor nastavenia.m*. Obsahuje desať premenných, trojicu pre každú z tried AF plus rýchlosť linky. Trojica premenných predstavuje veľkosť paketu, frekvenciu generovania týchto paketov a cieľovú adresu. Zadávané sú v nasledovnej podobe:

```
D1rate=600;
D2rate=200;      - počet generovaný paketov za sekundu
D3rate=150;
D1_size=8000;
D2_size=15000;  - veľkosť paketov v bitoch jednotlivých tried AF
D3_size=28000;
adresaAF1=1;
adresaAF2=2;     - adresa v zjednodušenom formáte na jedno číslo
adresaAF3=3;
linkspeed1=1000000;  - rýchlosť výstupnej linky na porte 1 v bitoch/s
```

Vkladanie je možné jednotlivo do *Command Window* Matlabu, potvrdením klávesou Enter je premenná uložená do *Workspace* alebo skopírovaním zo spomínaného súboru a znovu potvrdením klávesou Enter. Zmena hodnoty premenných je možná pred potvrdením jednoduchou editáciou rovnako ako v textovom editore, prípadne opakovaným zadaním premennej s inou hodnotou. Po zadaní týchto parametrov je možné spustenie simulácie.

Editácia pokročilejších nastavení ako sú profily WRED, klasifikácia paketov, smerovacia tabuľka a ďalšie, je potrebné previesť priamo vstupom do daného bloku a zmenou jeho nastavenia. Pomôckou pri prípadných problémoch môže byť popis blokov a ich nastavenia v predchádzajúcich podkapitolách.

Spustenie po zadaní hore uvedených hodnôt sa prevádza už priamo v okne Simulinku, tlačítkom *Start Simulation* v tvare symbolu play. Druhá možnosťou je otvoriť záložku na hornej lište *Simulation* a vybrať prvú položku *Start*. Dĺžka trvania simulácie sa dá nastaviť v okne *Simulation stop time*, napravo od tlačítka *Start*. Prednastavený čas je dvadsať sekúnd, ktorý postačuje na overenie všetkých parametrov. [7]

5. Vyhodnotenie výsledkov simulácie

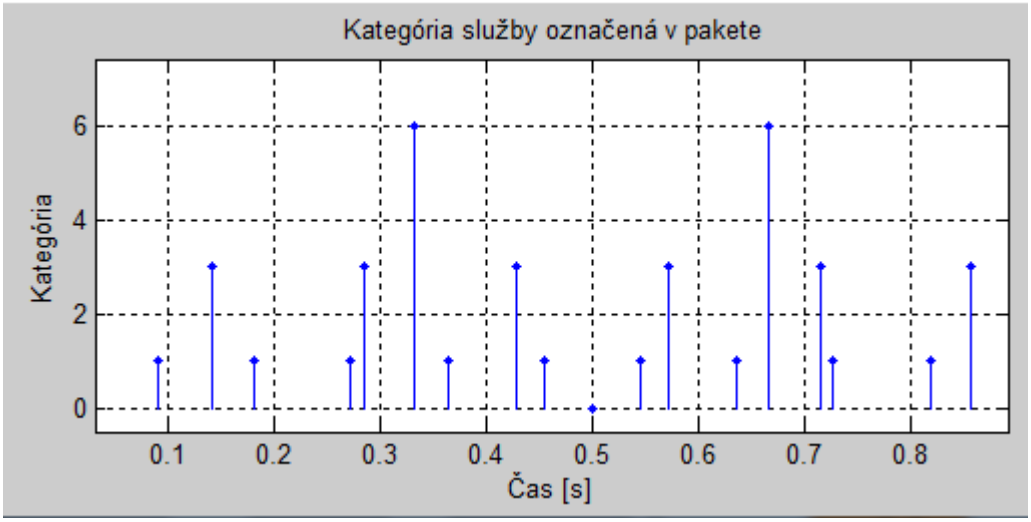
V tejto kapitole budú predstavené dosiahnuté výsledky a správanie jednotlivých mechanizmov bude porovnané s ich reálnym ekvivalentom. Simulácia bola zameraná na najperspektívnejšie technológie ako je aktívny manažment front WRED, spojovacie pole typu CBWFQ, správanie počas preskoku AF s využitím plánovača na predchádzanie kolíziám pri odosielaní paketov a ich riadený výber z front. Porovnanie niektorých parametrov ako napríklad oneskorenie nie je možné. Simulink aj pri výpočte, ktorý zaberie určitý čas nastavuje tento čas na nulu a tým pádom nedochádza v blokoch k žiadnemu oneskoreniu. Týmto spôsobom je kompenzovaná výkonnosť zariadenia, na ktorom je simulácia spúšťaná, ale nevýhodou je výsledná absencia týchto parametrov.

5.1 Klasifikácia paketov do tried AF

Činnosť klasifikátoru, teda bloku *PriradenieAFx,* je v zaradení prichádzajúceho paketu do jednej z tried AF, na základe jeho označenia kategórie služby, ktorú vyžaduje. Vnútorná štruktúra smerovača rozoznáva tri druhy tried AF. Kategórie služieb sú priradené k triedam AF nasledovne:

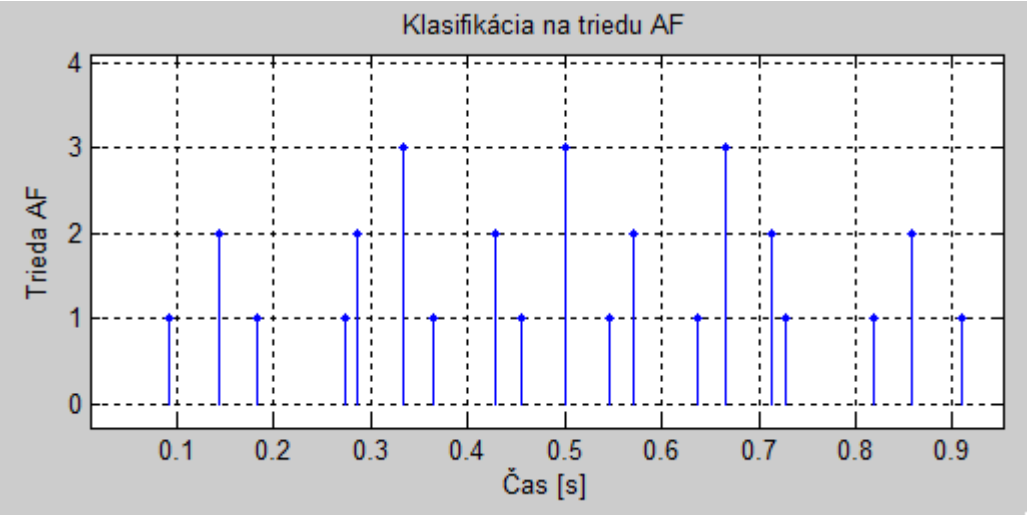
- AF1: kategórie 1 a 2
- AF2: kategórie 3, 4 a 5
- AF3: kategórie 6, 7, 8 a nevyznačená

Jednotlivými kategóriami sú označené služby podľa citlivosti na oneskorenie, stratovosť a kolísanie oneskorenia. Pod jednotkou a dvojkou je prenos hlasu cez dátové siete (VoIP) a videokonferencia. Pod kategóriami v triede AF2 prúdové a interaktívne služby a v triede AF3 služby na pozadí, ako FTP prenos a podobne. Činnosť bloku je znázornená na obrázkoch 5.1 a 5.2.



Obr. 5.1: *Kategórie prichádzajúcich paketov*

Body vynesené v rovnaký čas odpovedajú rovnakému paketu a z hodnoty vynesenej na osy *y* je možné sledovať správnu klasifikáciu. V čase pol sekundy prichádza na vstup neoznačený paket vyjadrený nulovou hodnotou, je prijatý a zaradený do triedy AF3.

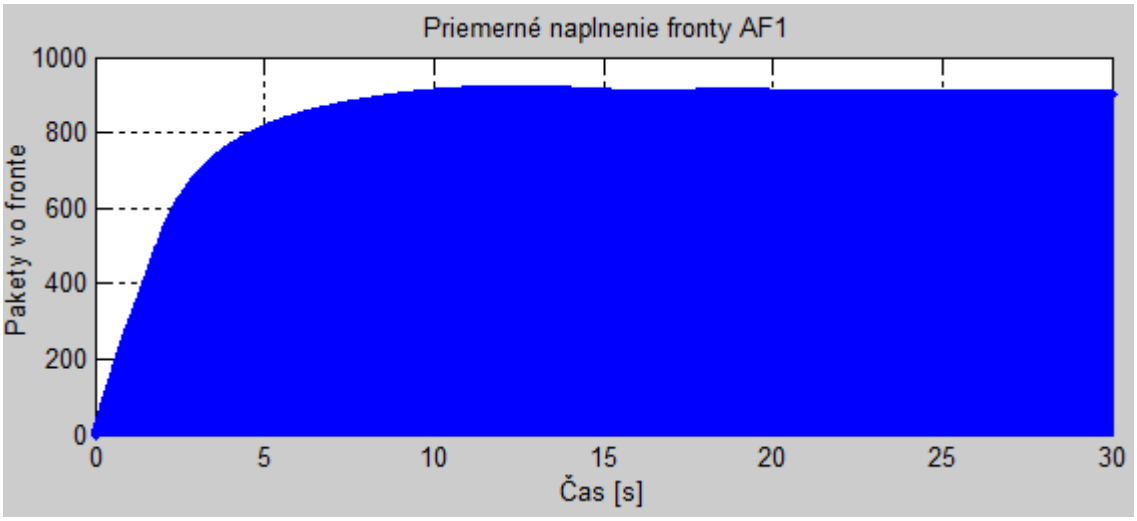


Obr. 5.2: *Výsledná klasifikácia paketov*

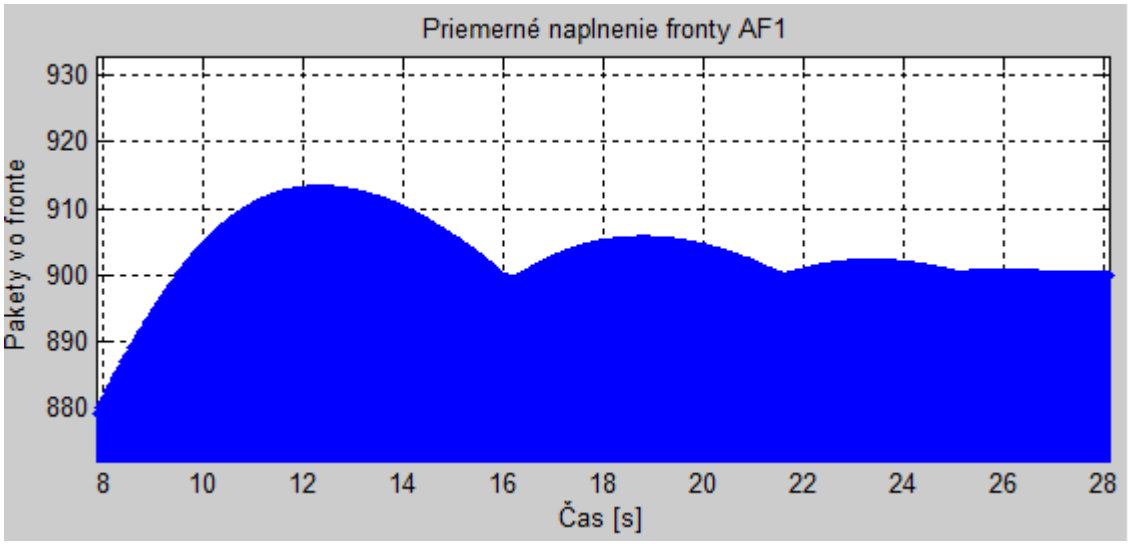
5.2 Dĺžka fronty WRED a príkazy na zahodenie

Aktívny manažment fronty je dôležitým prvkom pre efektívne využitie šírky pásma linky. Technológia WRED patrí medzi najnovšie v tejto skupine. Je nasadzovaná v najmodernejších sieťových prvkoch, o čom vypovedá jeho využitie v smerovačoch od spoločnosti Cisco v sériách 7000, 7500 a ďalších. Podrobne bola táto technika popísaná v predchádzajúcich kapitolách. Samostatné nastavenia profilov zahodenia pre každú triedu predstavujú v kombinácii s ďalšími opatreniami rozlíšenie služieb a priblíženie sa ich požiadavkám na kvalitu služieb (QoS). Nastavenie a správanie jednotlivých profilov v podobe naplnenia front v spojovacom poli pre triedy AF je nasledovné:

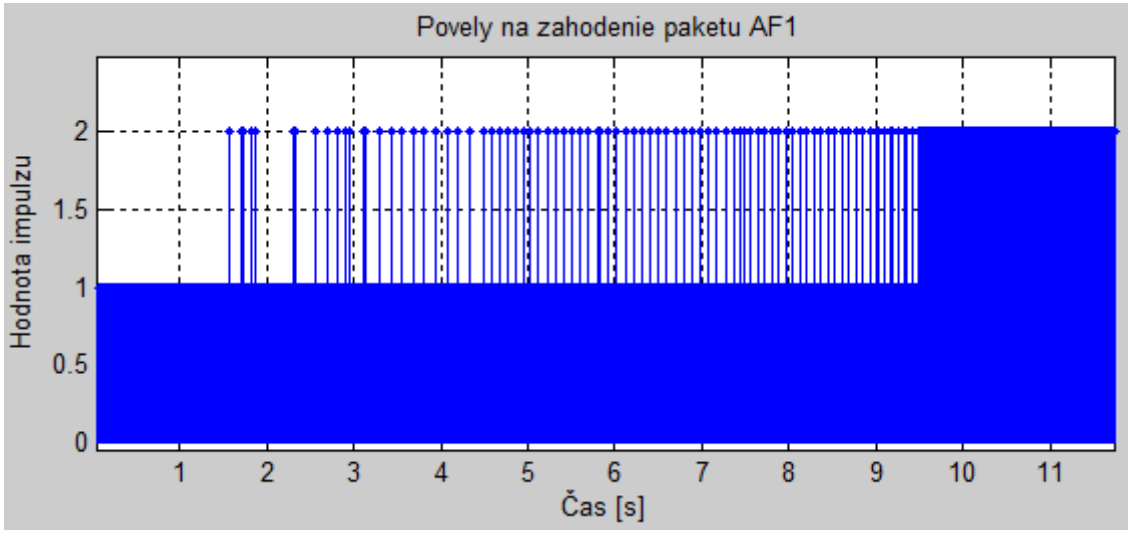
- Trieda AF1:
  Spodný prah α predstavujúci, kedy začína dochádzať k náhodnému zahadzovaniu je nastavený na α = 40%. Horný prah β je na úrovni β = 90%, kedy dochádza k zahadzovaniu každého prichádzajúceho paketu. Percentuálna hodnota vyjadruje priemerné zaplnenie fronty. Uhol nárastu pravdepodobnosti zahodenia paketu je rovný 30°. Kapacita fronty AF1 je tisíc paketov. Po naplnení fronty na úroveň horného prahu sa rast zastaví a hodnota začína oscilovať okolo úrovne prahu s postupným útlmom, až sa ustáli na hodnote horného prahu. Aktivácia WRED a spomalenie nárastu zaplnenia fronty je zobrazená na obrázku 5.3. Na priebehu je možné sledovať asymptotický charakter zaplnenia fronty so zrastajúcim percentom naplnenia. Tento jav je spôsobený zvyšujúcou sa pravdepodobnosťou zahodenia paketu, a teda stále viac zahodených prichádzajúcich paketov. Detailné zobrazenie oscilácie okolo horného prahu je zobrazené na obrázku 5.4. Narastanie počtu zahodených paketov je dobre pozorovateľné na zobrazení riadiacich impulzov z bloku WRED AF1, ktorý je zobrazený na obrázku 5.5. Hodnota riadiaceho impulzu jedna povoľuje vstup paketu a hodnota dva dáva pokyn na zahodenie paketu.



Obr. 5.3: *Vplyv WRED na plnenie fronty AF1*

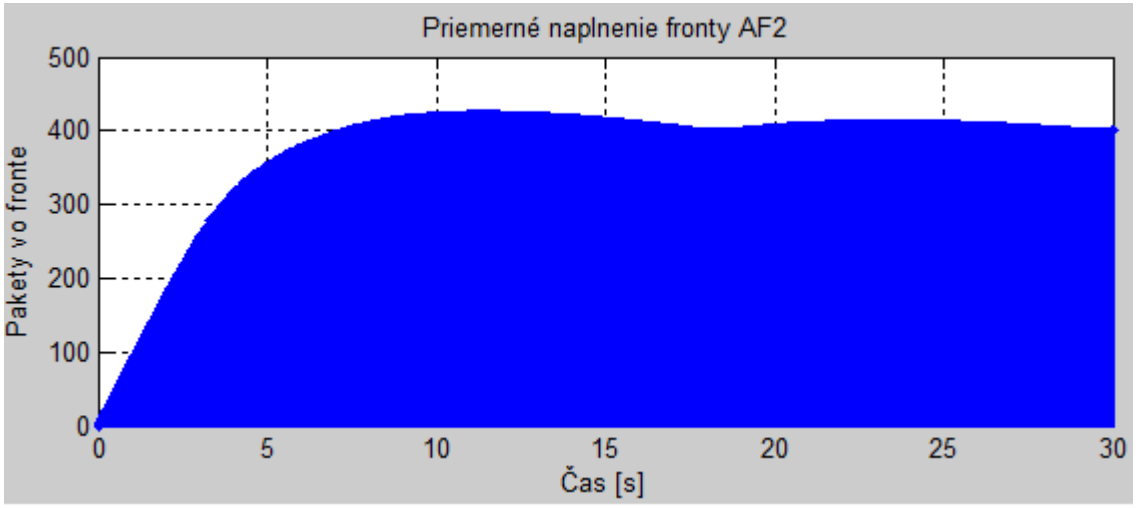


Obr. 5.4: *Oscilácia okolo horného prahu*

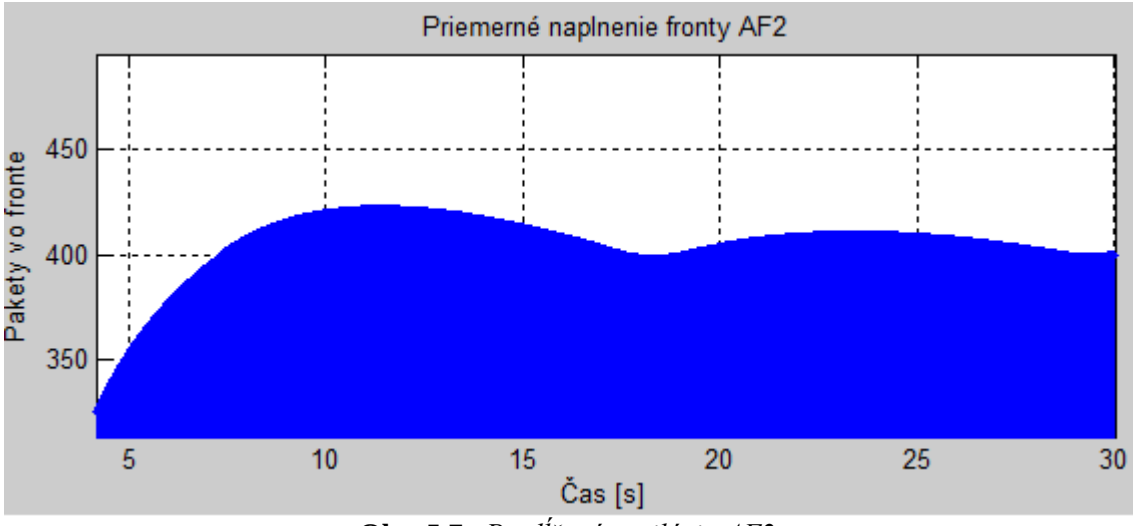


Obr. 5.5: *Riadenie zahadzovania*

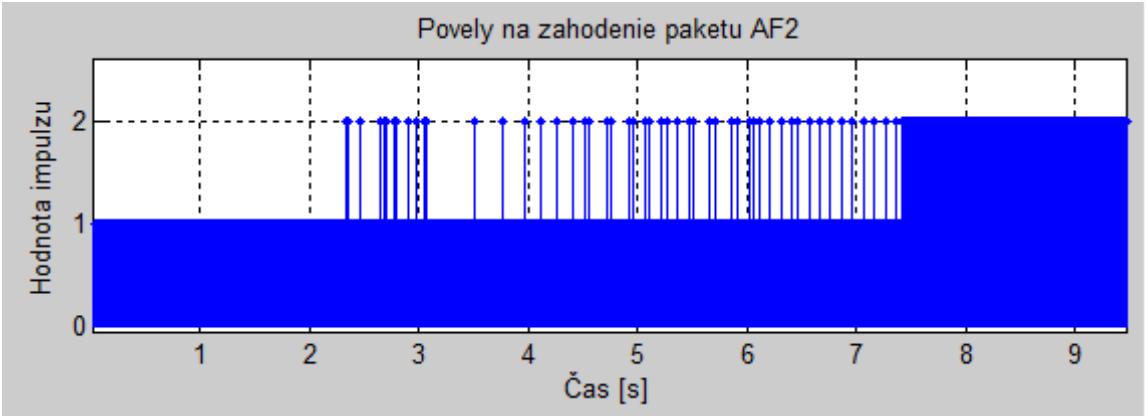
- Trieda AF 2:
  Hranica spodného prahu je nastavená na α = 35% a horný prah β = 80%. Veľkosť fronty je na úrovni päťsto paketov. Zníženie horného prahu a zväčšenie uhlu stúpania na 35° v kombinácii s nižšou rýchlosťou generovania paketov, má za následok zvýšený nárast pravdepodobnosti zahodenia a to sa prejaví pozvoľnejším nárastom zaplnenia fronty a zvýšením periódy oscilácie okolo horného prahu (Obr. 5.7).



Obr. 5.6: *Plnenie fronty AF2*

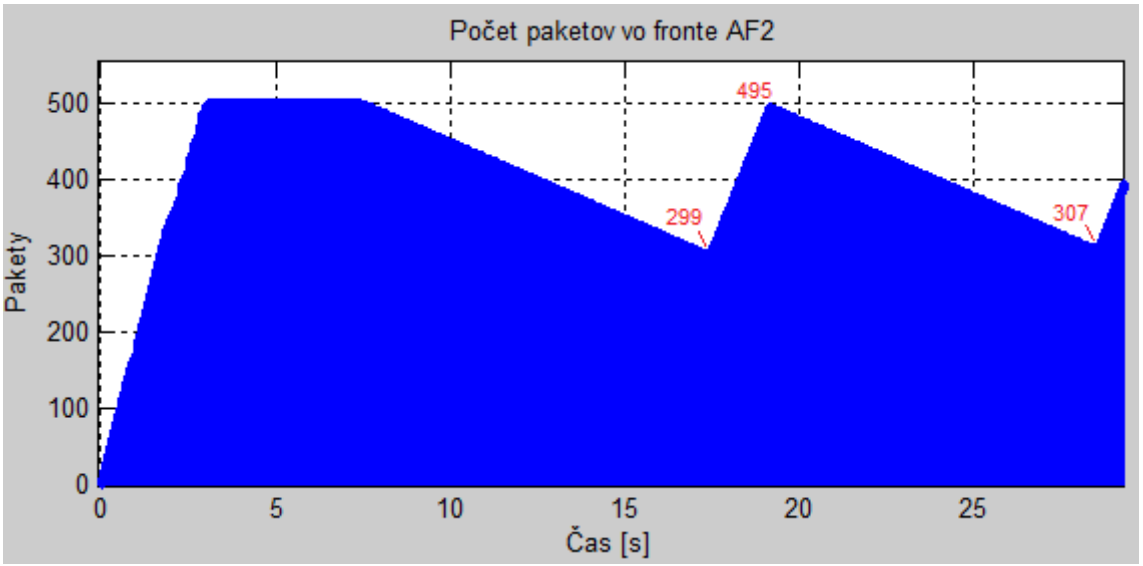


Obr. 5.7: *Predĺžená oscilácia AF2*

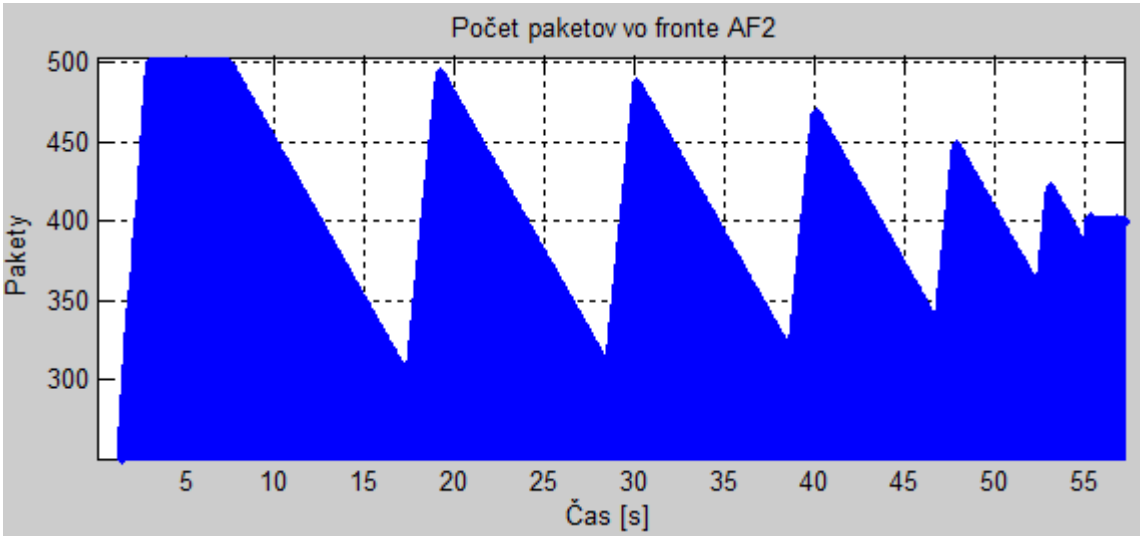


Obr. 5.8: *Jemnejší nárast zahadzovania*

Zaujímavé je sledovať zlepšovanie využitia fronty, s každou periódou oscilácie sa znižuje aj kolísanie aktuálneho naplnenia fronty. Tento jav je zobratený na obrázku 5.9. Po počiatočnom dosiahnutí zahltenia fronty, je počet paketov vo fronte znížený až na hodnotu dvestodeväťdesiatdeväť, následne dochádza znovu k narastaniu využitia fronty, ale k zahlteniu už nedôjde a plnenie sa zastaví na hodnote štyristodeväťdesiatštyri. Pri následnej oscilácii už neklesne využitie fronty tak rapídne. V dostatočne dlhom čase sa využitie ustáli na hodnote, ktorá odpovedá percentuálnemu nastaveniu horného prahu.

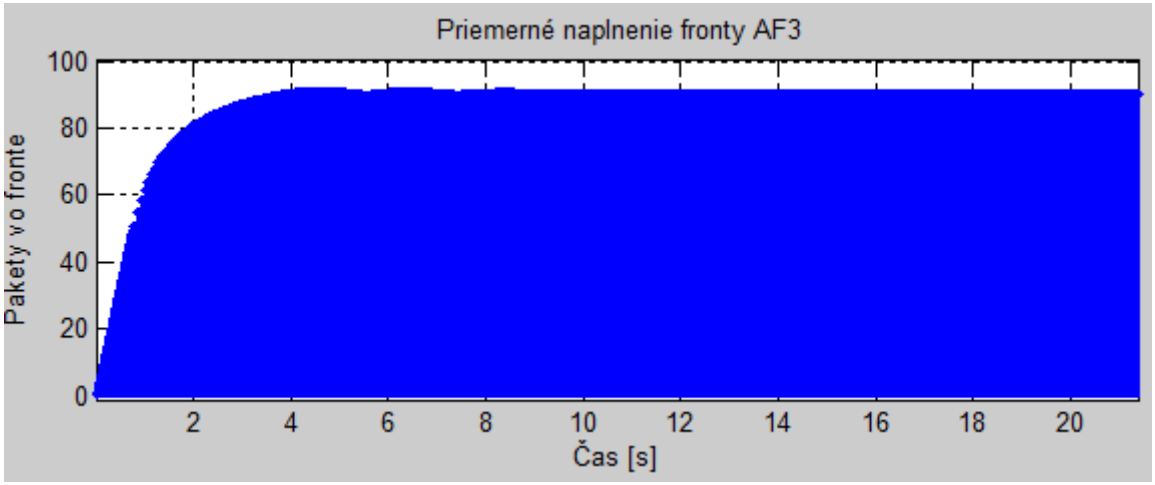


Obr. 5.9: *Správanie aktuálneho naplnenia fronty*

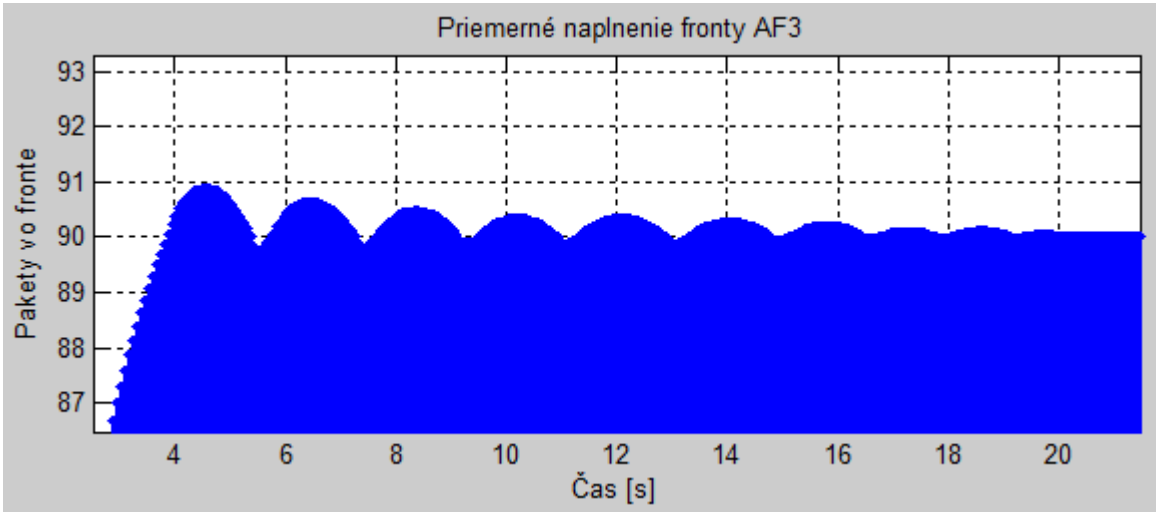


Obr. 5.10: *Ustálenie naplnenia v čase t*

- Trieda AF3:
  Táto trieda má najnižšiu prioritu, preto je spodný prah nastavený najnižšie a to na α = 30%, horný na β = 90% a uhol stúpania narástol na 45°. Veľkosť fronty pre tieto služby je nastavená na sto paketov. Pri takejto veľkosti fronty je dobre sledovateľný oscilačný jav aj v prípade zmeny o jeden paket.



Obr. 5.11: *Naplnenie fronty pre AF3*

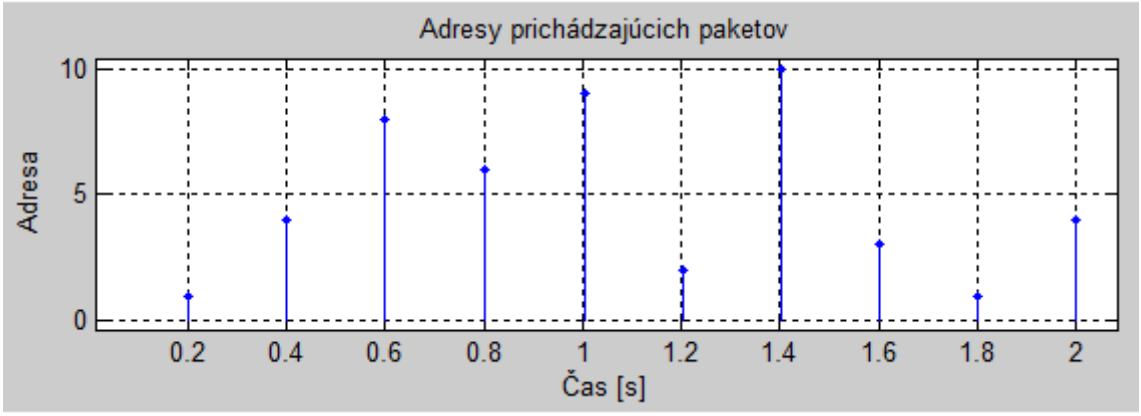


Obr. 5.12: *Detailné zobrazenie oscilácie*

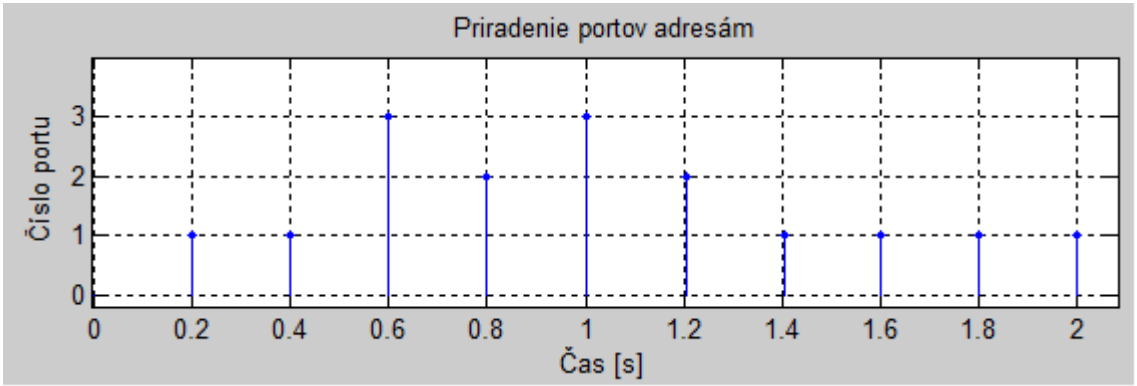
Výsledkom je zistenie závislosti medzi nastavením profilu zahadzovania a výškou a dĺžkou oscilácií. Vzdialenosť spodného a horného prahu zahodenia je nepriamo úmerná výške, a tým pádom aj dĺžke oscilácií. Veľkosť fronty na túto charakteristiku nemá vplyv. Pri nastavení prahov treba brať do úvahy aj ďalšie okolnosti, a to pri nastavení spodného prahu príliš nízko, môže dochádzať k zbytočnému zahadzovaniu paketov a teda nedostatočnému využitiu prenosovej linky. Ak je rozdiel medzi spodným a horným prahom príliš malý, môže dochádzať k globálnej TCP synchronizácii.

5.3 Spojovacie pole a smerovanie

Spojovacie pole slúži na prepájanie paketov na požadovaný výstupný port. K tomu slúži smerovacia tabuľka, v ktorej sú uložené adresy dosiahnuteľné na jednotlivých portoch. Pri príchode paketu je adresa z jeho záhlavia porovnaná v tabuľke a podľa toho je rozhodnuté o jeho prepnutí na príslušný port. Jednoduchý príklad smerovacej tabuľky použitej v simulácii je uvedený na obrázku 5.15.



Obr. 5.13: *Simulované adresy prichádzajúcich paketov*



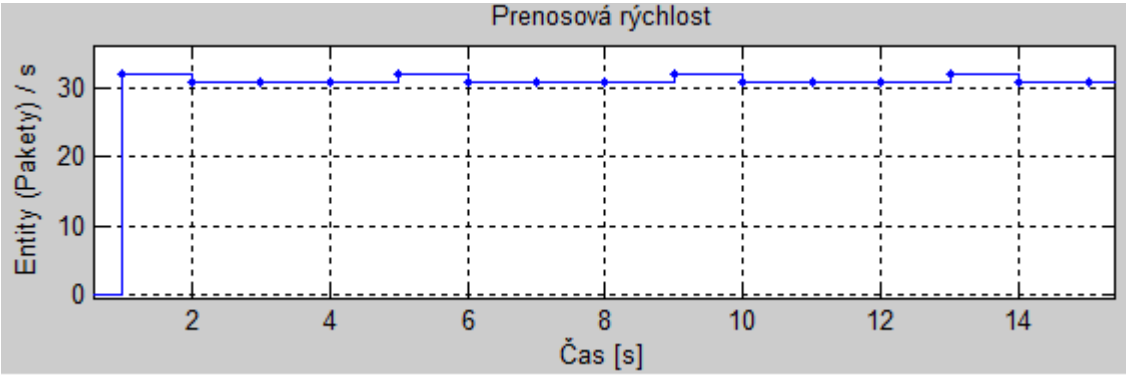
Obr. 5.14: *Preklad adries na porty smerovača*

	Adresy		
PORT 1	1	2	3
PORT 2	7	8	9
PORT 3	4	5	6
DEFAULT	1		

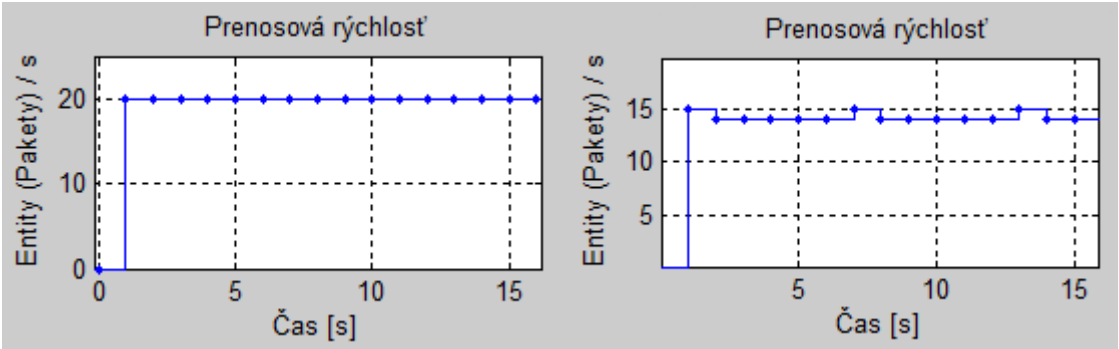
Tab. 5.1 *Použitá smerovacia tabuľka*

5.4 Plánovanie paketov a štatistiky výstupnej linky

Plánovanie výberu paketov z front je veľmi dôležité pre priblíženie sa požadovanej kvalite služieb. Pri výbere paketu rozhoduje jeho veľkosť a šírka pásma, ktorá je pre danú triedu AF vyhradená. Podľa pomeru veľkosti paketu k pridelenej šírke pásma sa určí poradie a pakety sú odosielané na výstupný port. Pri nemeniacej sa veľkosti paketov bude toto poradie zachované a bude sa opakovať počas celej doby prenosu (Obr. 5.15). V prípade, že po odoslaní paketov sa vyskytne prázdne okno, v ktorom dorazí jediný paket, je okamžite posielaný na výstup. Výsledkom plánovania je určitá paketová rýchlosť jednotlivých tried AF, ktorá by sa mala v bitovom vyjadrení blížiť hodnote vymedzenej šírky pásma. Tento fakt sa dá overiť jednoduchým výpočtom.



Obr. 5.15: *Paketová rýchlosť triedy AF1*



Obr. 5.16: *Paketová rýchlosť triedy AF2 v ľavo a AF3 v pravo*

Výpočet:

- Rýchlosť linky je 1Mb/s = 1 000 000 bitov z toho 95% je pridelených triedam AF

- Súčet bitových rýchlostí tried AF:

$$32 * 8000 = 256\,000$$

$$20 * 15\,000 = 300\,000$$

$$15 * 28\,000 = 420\,000$$

$$976\,000$$

Prenosová rýchlosť jednotlivých tried AF ako aj ich súčet sa blížia teoretickému predpokladu a dokazujú efektívne využitie kapacity prenosovej linky.

6. Záver

Hlavnou náplňou diplomovej práce je na základe teoretických znalostí vnútornej štruktúry sieťového prvku a jeho riadenia, navrhnúť a realizovať konštrukciu smerovača. Dosiahnuté výsledky podrobiť analýze a porovnaniu s reálnym sieťovým prvkom. V prvej kapitole sa pojednáva o požiadavkách kladených na sieťový prvok, hardvérových prostriedkoch a metódach ich realizácie. Podrobne sa venuje hlavne metódam prepojovania a obsluhy front. Popis techniky zabezpečenia kvality služieb DiffServ sa nachádza v druhej kapitole. Tretia kapitola je zameraná na problematiku smerovania za použitia neurónových sietí a ich vlastností. Praktická časť obsiahnutá v štvrtej kapitole sa venuje prostrediu Matlab/Simulink a predstave rozdelenia sieťového prvku na logické bloky a ich implementácii v programovom prostredí. Je navrhnutá štruktúra rozdelenia a popísaná funkcia každého bloku. Následne je detailne popísaná realizácia a zapojenie týchto blokov ako podsystémov. Súčasťou je zobrazenie ich zloženia z jednotlivých prvkov, ktoré poskytujú knižnice Simulinku, popis ich nastavenia a funkcie.

Realizovaná a odsimulovaná je konštrukcia sieťového prvku typu smerovač. Smerovač využíva klasifikáciu paketov, aktívny manažment fronty typu WRED, spojovacie pole typu CBWFQ s výberom výstupného portu podľa cieľovej adresy. Plánovanie paketov vzhľadom na ich veľkosť a veľkosť vyhradenej šírky pásma pre danú triedu AF. Pomocou jednoduchého zadania hodnôt v prostredí Matlab, je možné nastavovať základné parametre simulácie a sledovať správanie smerovača a implementovaných mechanizmov. Funkčnosť použitých algoritmov bola overená a výsledky sa blížia teoretickým hodnotám reálnych ekvivalentov. Bežné parametre ako oneskorenie a zmena oneskorenia, nie je možné porovnávať vzhľadom na simulačné vlastnosti prostredia Simulink. Oneskorenie by muselo byť vnášané umelo, čo je pre porovnanie výsledkov neakceptovateľné. Využitie linky pomocou plánovača paketov, predchádzanie zahlteniu pomocou aktívneho manažmentu front, klasifikácia prichádzajúcich paketov do tried AF, ako aj smerovanie, boli úspešne pozorované a vyhodnotené v piatej kapitole.

Použitá literatúra

[1] ZEDNIČEK, P. *Síťový prvek s pokročilým řízením*. Brno: Vysoké učení technické v Brne, Fakulta elektrotechniky a komunikačných technologií, 2010. 70 s.

[2] PARK, Kun I. *QoS in packet networks*. Boston : Springer Science + Bussines Media, Inc., 2005. 245 s. ISBN 0-387-23390-3.

[3] DTERG. *Dtreg.com* [online]. 2008 [cit. 2010-12-11]. Multilayer Perceptron Neural Networks. Dostupné z WWW: <http://www.dtreg.com>.

[4] DTREG. *Dtreg.com* [online]. 2008 [cit. 2010-12-11]. RBF Neural Networks. Dostupné z WWW: <http://www.dtreg.com>.

[5] JIRSÍK, V. *Neuronové sítě, expertní systémy a rozpoznávaní řeči Skripta 107 s*, Fakulta elektrotechniky a kominikačních techlologií VUT v Brně

[6] MOLNÁR, K. *Hardware počítačových sítí - propojovací uzly* [online], VUT Brno URL: <http://www.utko.feec.vutbr.cz/ molnar/index.php?stranka=bhws>.

[7] MATLAB 7.9.0 (R2009b). *Help and Documentation*

Zoznam použitých skratiek

ACK	Acknowledge
AF	Assured Forwarding
AQM	Active Queue Management
BA	Behavior Aggregate
CAR	Commited Acces Rate
CBQ	Class-Based Queuing
CBWFQ	Class-Based Weighted Fair Queing
CIR	Commited Information Rate
CSCP	Class Selector Code Points
CWR	Congestion Window Reduce
DiffServ	Differentiated Services
DSCP	DiffServ Code Points
ECN	Explicit Congestion Notification
EF	Expedited Forwarding
FIFO	First In First Out
FL	Flow Label
FR	Fair Queuing
IETF	Internet Enginnering Task Force
IP	Internet Protokol
NBAR	Network Based Application Recognition
PHB	Per Hop Behavior
PQ	Priority Queuing
QoS	Quality of Service
RBF	Radial Basis Function
RED	Random Early Discarding
RFC	Request for Comments
RIO	RED with the In/Out bit
SLA	Service Level Agreement
SYN	Synchronize
TC	Trafic Class
TCA	Traffic Conditioning Agreement
TCP	Transaction Control Protocol
ToS	Type of Service
UDP	User Datagram Protocol
VoIP	Voice over IP
VOQ	Virtual Output Aueues
WFQ	Weighted Fair Queuing
WRED	Weighted Random Early Discard
WRR	Weighted Round Robin

Zoznam príloh

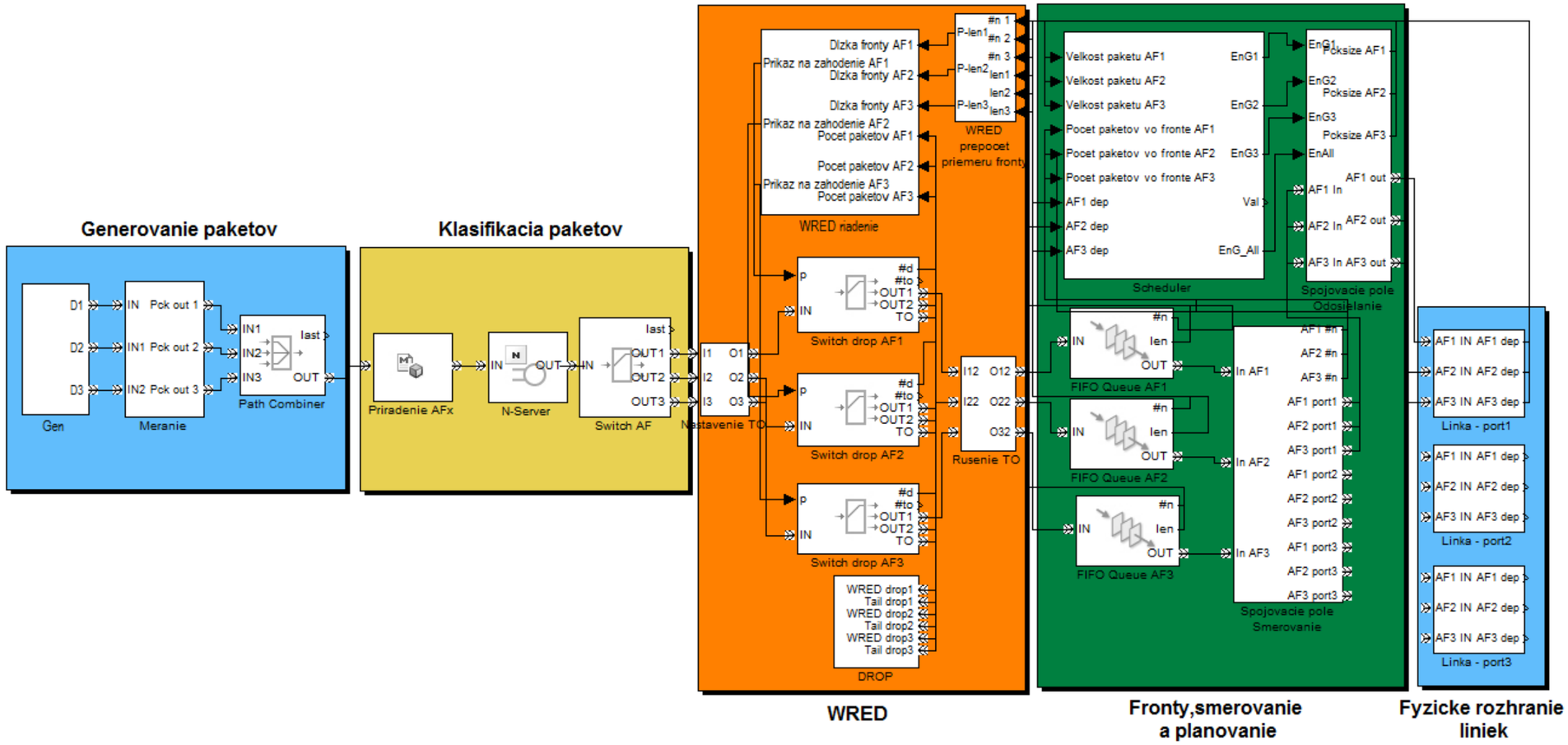
A. Zapojenie simulovaného sieťového prvku.

B. Schéma zapojenia bloku merania.

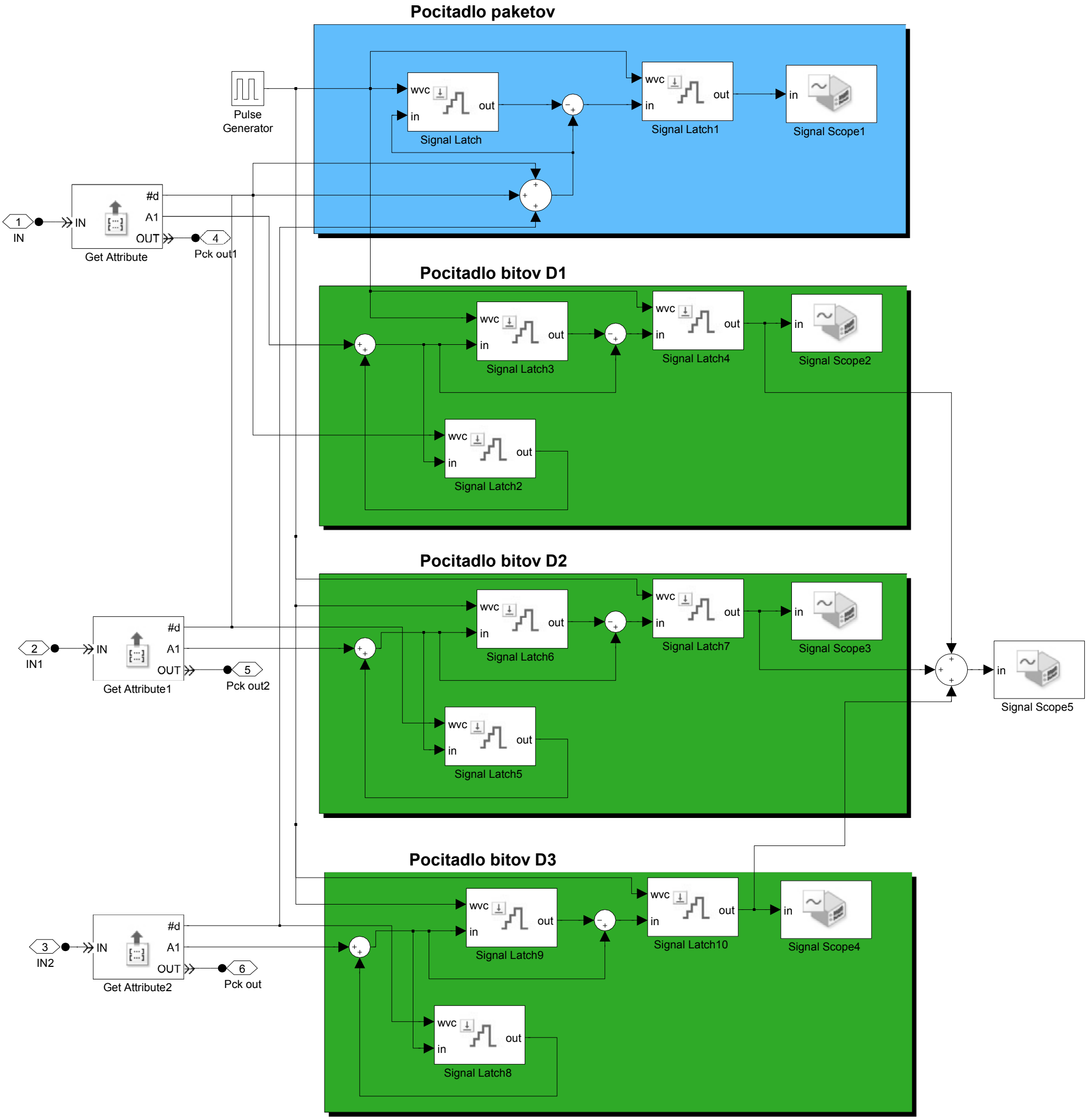
C. Schéma zapojenia bloku Scheduler.

D. DVD obsahujúce simuláciu a doplnkové súbory.
- obsahuje aj východzie obrázky v editovateľnom formáte (*MS Visio*)

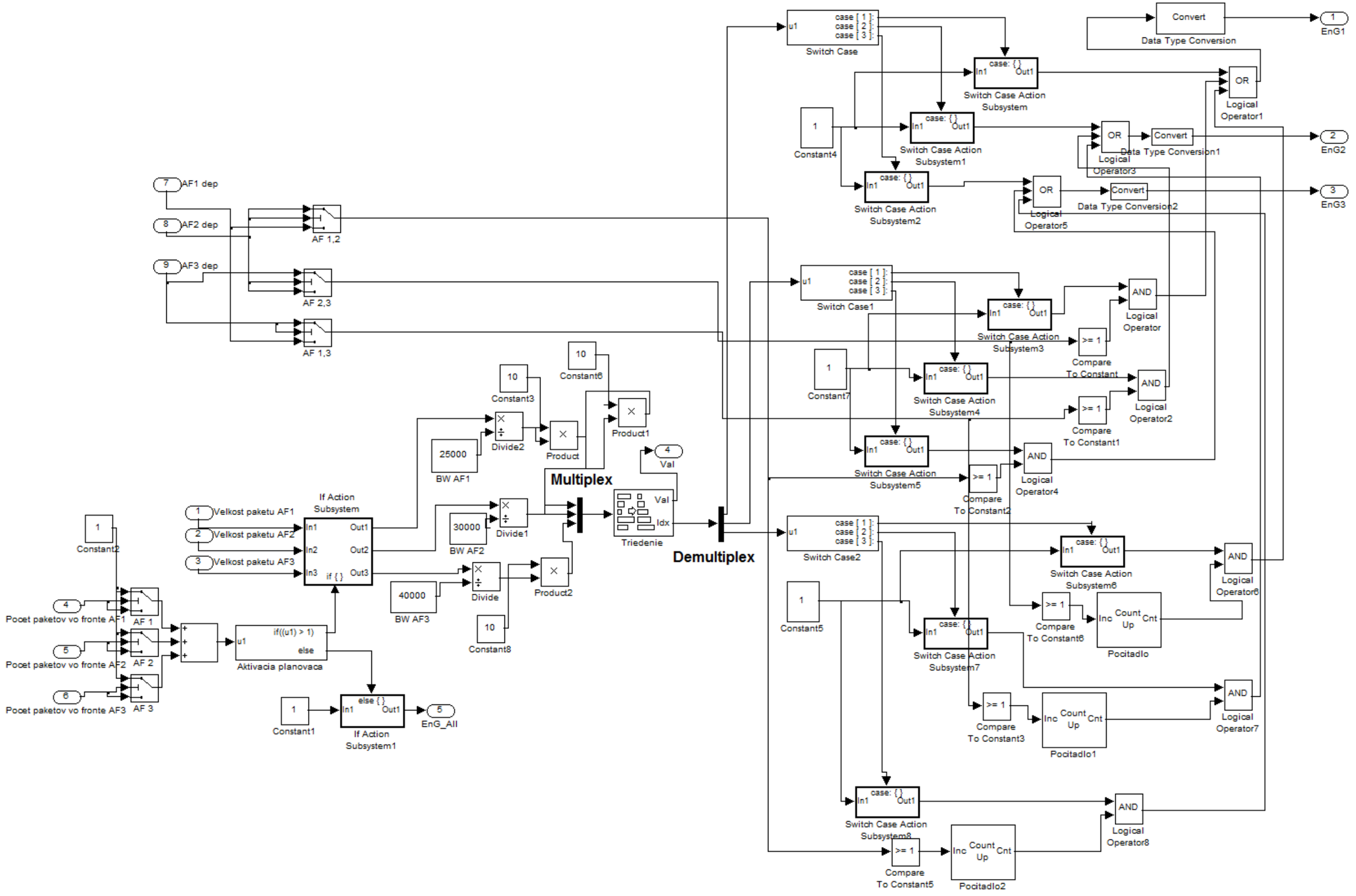
PRÍLOHA A


Generovanie paketov
Klasifikacia paketov
D1
D2
D3
IN
IN1
IN2
IN3
Pck out 1
Pck out 2
Pck out 3
last
OUT
Gen
Meranie
Path Combiner
Priradenie AFx
N-Server
Switch AF
I1
I2
I3
O1
O2
O3
Nastavenie TO
Dlzka fronty AF1
Prikaz na zahodenie AF1
Dlzka fronty AF2
Dlzka fronty AF3
Prikaz na zahodenie AF2
Pocet paketov AF1
Pocet paketov AF2
Prikaz na zahodenie AF3
Pocet paketov AF3
WRED nadenie
#n 1
#n 2
#n 3
P-len1
P-len2
P-len3
len1
len2
len3
WRED prepocet priemeru fronty
p
#d
#to
OUT1
OUT2
TO
Switch drop AF1
Switch drop AF2
Switch drop AF3
WRED drop1
Tail drop1
WRED drop2
Tail drop2
WRED drop3
Tail drop3
DROP
I12
I22
O12
O22
O32
Rusenie TO
Velkost paketu AF1
Velkost paketu AF2
Velkost paketu AF3
Pocet paketov vo fronte AF1
Pocet paketov vo fronte AF2
Pocet paketov vo fronte AF3
AF1 dep
AF2 dep
AF3 dep
EnG1
EnG2
EnG3
Val
EnG_All
Scheduler
Poksize AF1
Poksize AF2
Poksize AF3
EnAll
AF1 out
AF1 In
AF2 out
AF2 In
AF3 In AF3 out
Spojovacie pole
Odosielanie
#n
len
FIFO Queue AF1
FIFO Queue AF2
FIFO Queue AF3
In AF1
In AF2
In AF3
AF1 #n
AF2 #n
AF3 #n
AF1 port1
AF2 port1
AF3 port1
AF1 port2
AF2 port2
AF3 port2
AF1 port3
AF2 port3
AF3 port3
Spojovacie pole
Smerovanie
AF1 IN AF1 dep
AF2 IN AF2 dep
AF3 IN AF3 dep
Linka - port1
Linka - port2
Linka - port3
WRED
Fronty,smerovanie
a planovanie
Fyzicke rozhranie
liniek

PRÍLOHA B


Pocitadlo paketov
Pulse Generator
wvc
in
out
Signal Latch
Signal Latch1
Signal Scope1
1
IN
Get Attribute
#d
A1
OUT
4
Pck out1
Pocitadlo bitov D1
Signal Latch3
Signal Latch4
Signal Scope2
Signal Latch2
Pocitadlo bitov D2
2
IN1
Get Attribute1
5
Pck out2
Signal Latch6
Signal Latch7
Signal Scope3
Signal Latch5
Signal Scope5
Pocitadlo bitov D3
3
IN2
Get Attribute2
6
Pck out
Signal Latch9
Signal Latch10
Signal Scope4
Signal Latch8

PRÍLOHA C


7
AF1 dep
8
AF2 dep
9
AF3 dep
AF 1,2
AF 2,3
AF 1,3
10
Constant3
10
Constant6
25000
BW AF1
Divide2
Product
Product1
4
Val
Multiplex
If Action Subsystem
1
Velkost paketu AF1
2
Velkost paketu AF2
3
Velkost paketu AF3
In1
In2
In3
Out1
Out2
Out3
if { }
30000
BW AF2
Divide1
40000
BW AF3
Divide
Product2
10
Constant8
Triedenie
Val
Idx
Demultiplex
1
Constant2
4
Pocet paketov vo fronte AF1
5
Pocet paketov vo fronte AF2
6
Pocet paketov vo fronte AF3
AF 1
AF 2
AF 3
if((u1) > 1)
u1
else
Aktivacia planovacka
1
Constant1
else { }
If Action Subsystem1
5
EnG_All
Switch Case
case [1]:
case [2]:
case [3]:
Switch Case Action Subsystem
Switch Case Action Subsystem1
Switch Case Action Subsystem2
1
Constant4
Convert
Data Type Conversion
1
EnG1
OR
Logical Operator1
OR
Logical Operator3
Convert
Data Type Conversion1
2
EnG2
OR
Logical Operator5
Convert
Data Type Conversion2
3
EnG3
Switch Case1
Switch Case Action Subsystem3
AND
Logical Operator
>= 1
Compare To Constant
1
Constant7
Switch Case Action Subsystem4
AND
Logical Operator2
>= 1
Compare To Constant1
Switch Case Action Subsystem5
AND
Logical Operator4
>= 1
Compare To Constant2
Switch Case2
Switch Case Action Subsystem6
AND
Logical Operator6
1
Constant5
Switch Case Action Subsystem7
>= 1
Compare To Constant6
Inc
Count Up
Cnt
Pocitadlo
>= 1
Compare To Constant3
Inc
Count Up
Cnt
Pocitadlo1
AND
Logical Operator7
Switch Case Action Subsystem8
AND
Logical Operator8
>= 1
Compare To Constant5
Inc
Count Up
Cnt
Pocitadlo2