ČESKÉ VYSOKÉ UČENÍ TECHNICKÉ V PRAZE

Fakulta elektrotechnická

Katedra telekomunikační techniky

Metody omezování a tvarování toku v IP síti

květen 2015

Bakalant: Tomáš Krbec

Vedoucí práce: Ing. Petr Hampl, Ph.D.

Čestné prohlášení

Prohlašuji, že jsem zadanou bakalářskou práci zpracoval sám s přispěním vedoucího práce a konzultanta a používal jsem pouze literaturu v práci uvedenou. Dále prohlašuji, že nemám námitek proti půjčování nebo zveřejňování mé bakalářské práce nebo její části se souhlasem katedry.

Datum: 22. 5. 2015 .....

podpis bakalanta

Poděkování:

Tímto bych chtěl poděkovat svému vedoucímu bakalářské práce Ing. Petru Hamplovi, Ph.D. za jeho odborné vedení, trpělivost a ochotný přístup při konzultování této bakalářské práce. Dále bych chtěl poděkovat své přítelkyni Bc. Barbaře Levkové za její cenné rady při úpravě textu.

Anotace:

Tato bakalářská práce se zabývá metodami omezování a tvarování toků v IP síti, kterému je také věnována část bakalářské práce včetně názorné ukázky tří simulačních scénářů v simulačním programu OMNeT++. Převážná část obsahu praktické části se především zaměřuje na Weighted Fair Queueing a jeho navržení v simulačním programu.

Klíčová slova: OMNeT++, Weighted Fair Queueing, omezování, tvarování toku

Summary:

This bachelor thesis deals with traffic shaping and policing in the OMNeT++ environment including three scenarios in this simulation environment. Most of the practical part is focused on Weighted Fair Queueing and its design in OMNeT++.

Index Terms: OMNeT++, Weighted Fair Queueing, traffic, shaping, policing

Seznam zkratek

QoS	Quality of Service
BE	Best-Effort
Intserv	Integrated Services
Diffserv	Differentiated Services
IP	Internet Protocol
ToS	Type of Service
DTR	Delay, Throughput, Reliability
MBZ	Must Be Zero
DS	Differentiated Services
ECN	Explicit Congestion Notification
PIR	Peak Information Rate
CIR	Committed Information Rate
CBS	Committed Burst Size
EBS	Excess Burst Size
PBS	Peak Burst Size
srTCM	Single Rate Three Color Marker
trTCM	Two Rate Three Color Marker
PQ	Priority Queueing
WFQ	Weighted Fair Queueing
CQ	Custom Queueing
CDT	Congestive Discard Threshold
HQL	Hold Queue Limit
FIFO	First In First Out
SLA	Service Level Agreement
AQM	Active Queue Management
DSCP	Differentiated Services Code Point
HD	High Definition
UDP	User Datagram Protocol
TCP	Transmission Control Protocol

Obsah

Úvod

Omezování a tvarování toku je v současné době stále aktuálním tématem, které mě oslovilo, a proto jsem si vybral toto téma bakalářské práce. Příkladem k současnému dění může být nekalé využití omezování toku, ke kterému došlo v roce 2014, kdy se řešil spor velkých amerických poskytovatelů internetu se společností Netflix poskytující videoobsah ve vysokém rozlišení. Poskytovatelé, přes které procházel obsah Netflixu ke koncovým zákazníkům, tento obsah omezovali, dokud nedošlo k finančnímu vyrovnání se společností Netflix. Tato bakalářská práce je rozdělena na dvě části: teoretickou část a simulační část s připravenými scénáři.

V první kapitole teoretické části čtenářům nejdříve vysvětlím, co se děje s paketem po příchodu na vstupní rozhraní směrovače, jeho klasifikaci a jaké procesy mohou ovlivnit průchod paketu sítí. V dalších dvou kapitolách přiblížím metody omezování intenzity toku a upřednostnění paketů v případě zahlcení.

V praktické části popisuji použitý simulační program OMNeT++ a tři mnou vytvořené simulační scénáře pro porovnání režimu front a potvrzení předpokladu reakce na konkrétní druh síťového provozu. Převážná část obsahu praktické části se především zaměřuje na Weighted Fair Queueing, který byl po dohodě s vedoucím mé bakalářské práce sestaven dle vlastního návrhu pouze s jednou pamětí.

Již zmíněné scénáře budou sestaveny pro zjištění reakce na konkrétní druh síťového provozu. Všechny scénáře budou pracovat se stejným generovaným tokem, který bude vygenerován na transportní vrstvě ISO/OSI modelu. Při interpretaci výsledků budu klást zvýšenou pozornost na zpoždění paketů a počet ztracených paketů. V závěru mé bakalářské práce budou výsledky simulací statisticky zpracovány s hladinou významnosti a konfidenčním intervalem.

1 Kvalita služby (QoS)

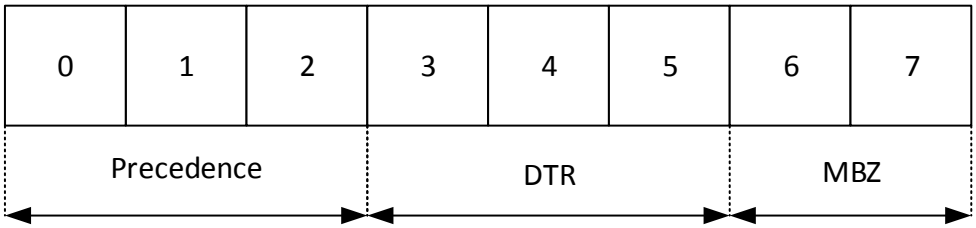
S rozmachem telekomunikačních sítí a jejich společné konvergenci, konkrétně datových a hlasových služeb, do jednoho přenosového kanálu je potřeba řešit problematiku „úzkého hrdla“ a omezit či vyhradit přenosové pásmo nebo upřednostnit jeden datový provoz před druhým. A u určitých služeb řešit i například včasnost doručení.

V IP paketových sítích jsou pro zajištění odpovídající kvality služby QoS používány následující modely:

- Best-Effort (BE) – síť negarantuje doručení do cíle,
- Integrated Services (Intserv) – dochází k rezervaci přenosové kapacity od počátečního uzlu až po koncový uzel, použitelné v rámci jednoho autonomního systému,
- Differentiated Services (Diffserv) – pakety jsou v rámci diffserv domény agregovány do několika tříd.

1.1 Značení paketů (Marking)

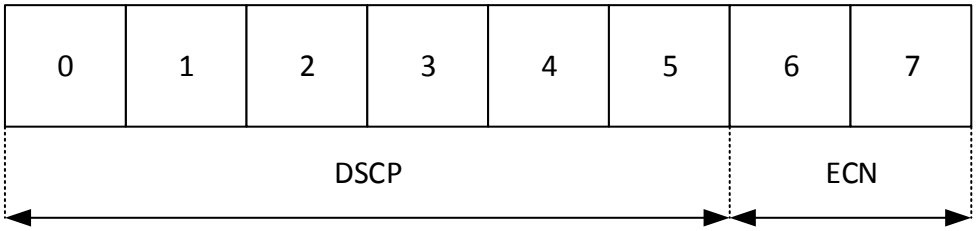
Pro Diffserv model dochází k přeznačkování na hraničním směrovači (Edge/Border směrovač) při vstupu do Diffserv domény. Označení může například provádět i IP telefon v lokální IP síti. Klíčové je, zda lze značce důvěřovat. Pokud nelze značce důvěřovat dochází po příchodu paketu na rozhraní směrovače nejprve k značení paketů. To probíhá na třetí vrstvě ISO/OSI modelu, kde se nalézá v záhlaví IP paketu pole označované jako Type of Service (ToS). Na obr. 1.1 lze vidět dřívější strukturu tohoto pole.



Obr. 1.1 - Dřívější značení pole ToS

První tři bity zabírala hodnota IP Precedence, která dosahovala hodnot v intervalu od 0 do 7 pro nastavení priority paketu. Tříbitová hodnota DTR (Delay, Throughput, Reliability) sloužila jako ukazatel pro způsob průchodu sítí. Sledované

parametry průchodu sítí byly např. propustnost nebo zpoždění. Poslední dva bity MBR (Must Be Zero) se nastavovaly na nulovou hodnotu [1].
Aktuální použití pole ToS v záhlaví IP paketu můžeme vidět na obr. 1.2.



Obr. 1.2 - Nynější značení pole ToS

IP Precedence a následující hodnota ToS byly nahrazeny DSCP značkou používající se v Diffserv. Zpětná kompatibilita IP precedence zůstala v DSCP zachována. Avšak každá DSCP značka se chová nezávazně v různých DS doménách. Poslední dva bity ECN slouží pro indikaci zahlcení sítě [2].

1.2 Klasifikace paketů (Classification)

Dalším procesem, kterým prochází paket, je rozřazení podle předem definovaných pravidel do prioritních tříd. Klasifikace může probíhat na základě DSCP značky, ale i např. zdrojového či cílového portu nebo rozhraní.

1.3 Omezování toku (Traffic Policing)

Dalším důležitým krokem při dodržování kvality služeb je kontrola toků, zdali dodržují například smluvně stanovené přenosové rychlosti.
V rámci omezování toku, zohledňujeme dvě přenosové rychlosti:

- **špičková přenosová rychlost (PIR – Peak Information Rate)** – maximální přenosová rychlost přenášená na síťové rozhraní, udávaná v bajtech za sekundu,
- **průměrná přenosová rychlost (CIR – Committed Information Rate)** – smluvně dohodnutá přenosová rychlost, udávaná v bajtech za sekundu.

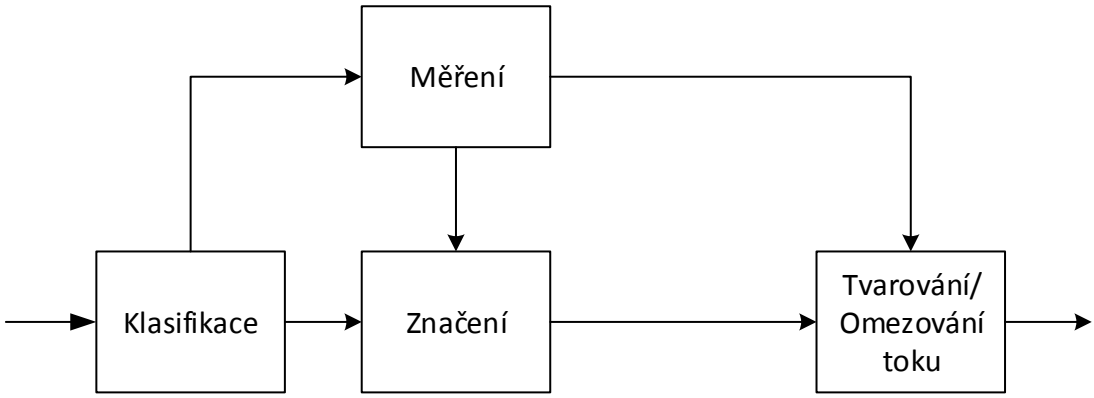
Při počítání špičkové a průměrné přenosové rychlosti se zpravidla započítává pouze záhlaví třetí vrstvy ISO/OSI modelu. Záhlaví nižších vrstev není v těchto rychlostech zohledněno.

Dále pro upřesnění omezení toku rozlišujeme tyto pomocné parametry:

- **průměrná velikost shluku paketů (CBS – Committed Burst Size)** – garantovaná maximální velikost shluku paketů v bajtech, které můžou být přeneseny ve špičkové přenosové rychlosti tak, aby zároveň splňovaly přenos s průměrnou přenosovou rychlostí,
- **nadměrná velikost shluku paketů (EBS – Excess Burst Size)** – po překročení této shlukové velikosti jsou pakety většinou zahozeny,
- **špičková velikost shluku paketů (PBS – Peak Burst Size)** – velikost funkcí podobná nadměrné shlukové velikosti využívající se ve spojení se špičkovou přenosovou rychlostí.

1.4 Tvarování toku (Traffic shaping)

Tento pojem bývá často porovnáván s omezováním toku z hlediska funkcionality, ale při tvarování toku se jedná především o maximální využití přenosové kapacity.



Obr. 1.3 – Blokové schéma QoS ve směrovači

Na obr. 1.3 lze vidět, že nejprve dochází ke klasifikaci paketů a dále se způsob průchodu paketu liší podle použití. Pokud by například docházelo k tvarování toku na výstupním rozhraní, tak v bloku měření by byl jeden z přístupů omezující intenzitu toků a bloku tvarování jeden z přístupů upřednostnění paketů.

Tyto přístupy budou zmíněny v následujících kapitolách.

Při tvarování toku se využívá předpoklad, že během krátkého časového intervalu dochází k velkým rozdílům potřebné přenosové kapacity až nad maximální možnou přenosovou kapacitu následovanou pádem na hodnotu, kdy linka je téměř nevyužita. Pakety, které přesahují maximální možnou přenosovou kapacitu (nebo nastavenou hraniční hodnotu), jsou uchovány v paměti (bufferu) směrovače a po

uvolnění linky odeslány.

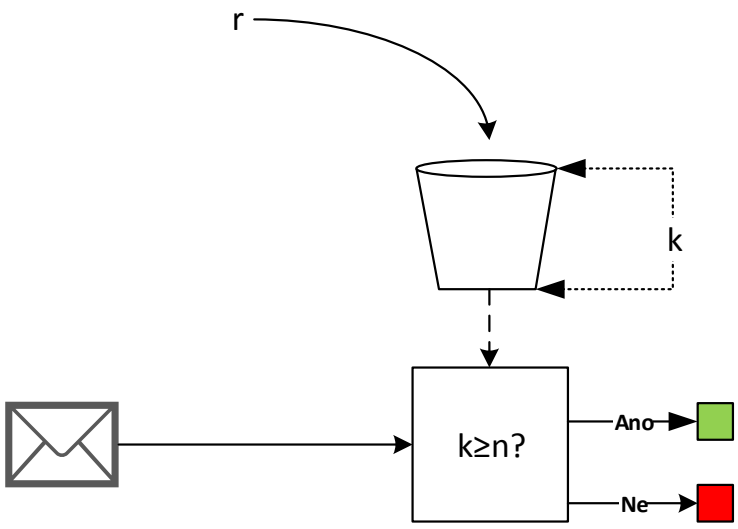
Nevýhodou tohoto postupu je, že při nabízeném zatížení nad maximální přenosovou kapacitu po delší dobu může dojít k přetečení paměti a zahazování paketů. Další nevýhodou je zvyšování zpoždění a jeho rozptylu (variation of delay).

2 Přístupy omezující intenzitu toků

V předchozí kapitole (1.4) bylo zmíněno, že k omezení toku většinou dochází na vstupním rozhraní. Jedním z příkladů může být omezování zákazníků dle sjednaného tarifu na agregované lince, aby nenastala situace, kdy je přenosová kapacita využívána pouze jedním uživatelem bez ohledu na druhé.

V následujících podkapitolách budou popsány některé ze základní metod k omezení intenzity toků.

2.1 Token Bucket



Obr. 2.1 – Analogie Token bucket

Tento přístup lze popsat jako analogii vědra, do kterého přitékají tokeny rychlostí r [B/s].

$$r \approx CIR \tag{2.1}$$

Střední doba mezi příchody tokenů se vypočítá pomocí následující rovnice.

$$\Delta T_r = \frac{1}{r} \text{ [s]} \tag{2.2}$$

Každý token má hodnotu jednoho bajtu. Vědro má kapacitu tokenů k. Po dosažení maximální kapacity tokeny nepřibývají, pouze „přetékají“ do té doby než se ve vědru uvolní místo.

$$k \approx CBS \tag{2.3}$$

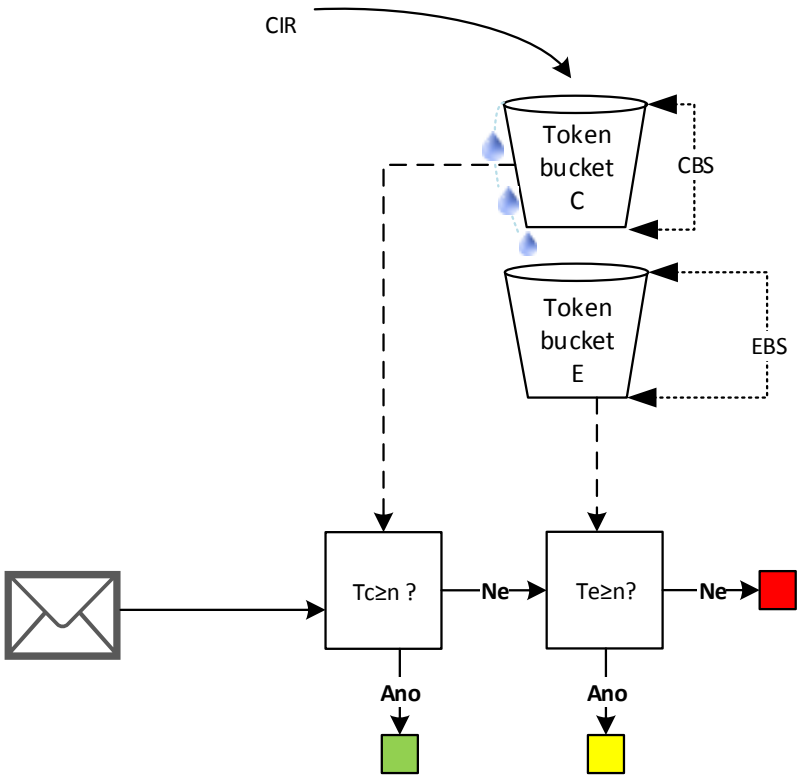
Pokud na vstup dorazí paket o velikosti n, algoritmus zjistí velikost paketu a dle Obr. 2.1 mohou nastat dvě situace:

1. Velikost paketu je menší než počet tokenů ve vědru, paket je označen jako vyhovující (na obr. 2.1 zeleně) a odeslán. Z vědra je odebráno *n* tokenů, kde *n* je velikost paketu v bajtech.
2. Velikost paketu je větší než počet tokenů. Paket je označen jako nevyhovující (na obr. 2.1 červeně) a podle použití popř. i zahozen.

Výše popsané chování je při chování metody Token Bucket jako omezovač toku (viz 1.3). Při použití jako tvarovače toku (viz 1.4) při nedostatečném počtu tokenů k průchodu paketu je paket pozdržen v paměti do té doby, než vědro bude disponovat dostatečným počtem tokenů [3].

2.2 Single Rate Three Color Marker (srTCM)

Cílem toho omezovače je zaručit, aby z dlouhodobého hlediska byl uživatelův průměrný tok podobný průměrné přenosové rychlosti (CIR) [4].



Obr. 2.2 – Základní analogie srTCM

Jak již název napovídá, tok je řízen na základě jedné přenosové rychlosti. Na obr. 2.2 je vidět, že omezovač se skládá z dvou samostatných věder C a E o rozdílných kapacitách CBS a EBS závislých na shlukové velikosti paketů.

Nejprve tokeny nabývají do prvního vědra C

$$T_c < CBS \rightarrow T_c = T_c + 1, \tag{2.4}$$

a to rychlostí CIR (tj. každou 1/CIR). Pokud je vědro C zaplněno, tak tokeny „přetékají“ do vědra E. Při inicializaci jsou obě vědra zaplněna, aktuální zaplnění věder se značí T_c a T_e.

Tokeny mají opět hodnotu jednoho bajtu, dle obr. 2.2 mohou nastat následující situace:

1. Paket o velikosti *n* bajtů dorazí na vstup srTCM a splňuje podmínku

$$T_c - n \geq 0 \,, \tag{2.5}$$

   paket je označen zeleně a zaplnění vědra je sníženo o velikost paketu

$$T_c = T_c - n \ . \tag{2.6}$$

2. Paket o velikosti *n* bajtů dorazí na vstup srTCM a splňuje následující podmínky

$$(T_c - n < 0) \wedge (T_e - n \geq 0) \,, \tag{2.7}$$

   paket je označen žlutě a zaplnění druhého vědra je sníženo o velikost paketu

$$T_e = T_e - n \ . \tag{2.8}$$

3. Paket o velikosti *n* bajtů dorazí na vstup srTCM a kapacita věder není pro jeho odeslání dostatečná a paket je označen červeně

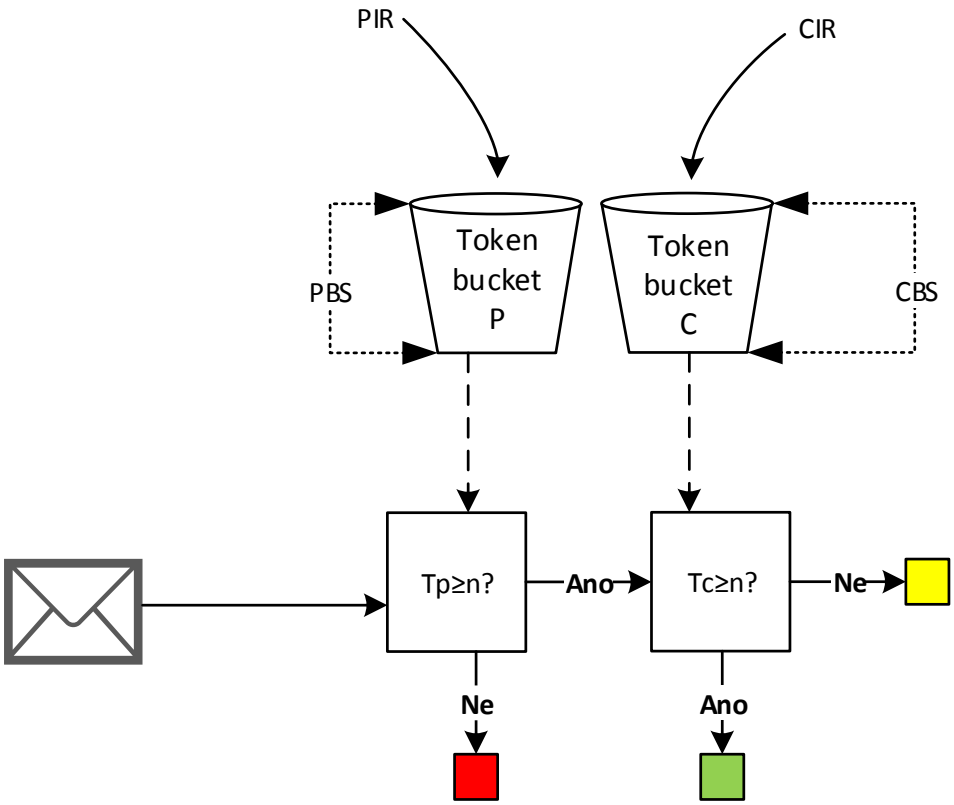
$$(T_c - n < 0) \wedge (T_e - n < 0) \ . \tag{2.9}$$

Další důležitou věcí je, že tento omezovač může fungovat ve dvou módech:

- Slepý mód (Color-Blind Mode) – kdy se nerozlišuje, s jakou barvou přišel paket na vstup, funkce popsána výše.
- Mód vnímající barvu (Color-Aware Mode) – omezovač rozlišuje vstupní barvu paketu:
  - paket přijde jako zelený, je s ním zacházeno stejně jako ve slepém módu,
  - paket přijde jako žlutý, v situaci 1 a 2 vychází jako žlutý. Při nedostatečné kapacitě věder je označen jako červený,
  - paket přijde jako červený, ihned vystupuje jako červený.

2.3 Two Rate Three Color Marker (trTCM)

Na rozdíl od předchozího omezovače, tento je závislý na dvou přenosových rychlostech. Do věder P a C jsou tokeny doplňovány různou rychlostí.



Obr. 2.3 – Základní analogie trTCM

V prvotním okamžiku obě vědra disponují maximem tokenů

$$\begin{aligned} T_p(0) &= PBS \\ T_c(0) &= CBS \end{aligned} . \tag{2.10}$$

Na obr. 2.3 si lze povšimnout, že při průchodu paketu dochází nejprve k porovnání velikosti paketu *n* se zaplněním vědra P. Pokud není splněna podmínka

$$T_p - n \geq 0 , \tag{2.11}$$

paket je označen červeně.

Při splnění jsou tokeny z vědra P odečteny a otestována následující podmínka

$$\begin{aligned} T_p &= T_p - n \\ T_c - n &\geq 0 \end{aligned} . \tag{2.12}$$

Na výběr už jsou pouze jen dvě možnosti, buď bude disponibilní kapacita vědra C nedostatečná a paket bude označen žlutě nebo dostatečná a paket s odečtením patřičných tokenů bude označen zeleně

$$T_c = T_c - n \ . \tag{2.13}$$

Stejně jako u srTCM (viz 2.2) je i zde možnost funkce v módech. Chování je naprosto stejné, u přicházejícího zeleného je funkcionalita popsána výše, žlutý může po průchodu omezovačem zůstat v nejlepším případě jako žlutý a na červený se nebere ohled, zůstává stejný.

3 Přístupy upřednostnění paketů v případě zahlcení

Zahlcení je nežádoucí jev objevující se v sítích. Ale pokud nastane, lze důležité služby na úkor ostatních služeb zachovat. Mezi základní přístupy patří například:

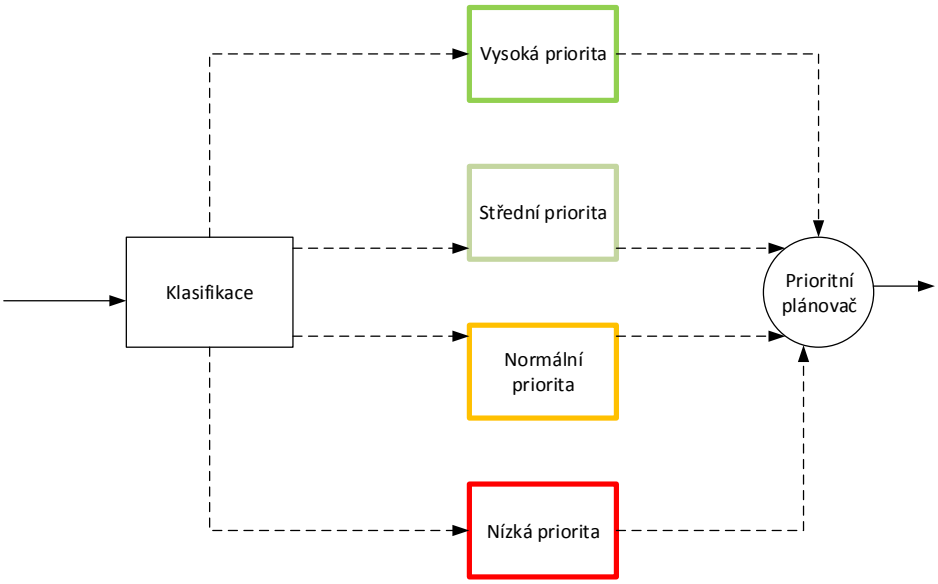
- Priority Queuing (PQ),
- Weighted-Fair Queueing (WFQ),
- Custom Queueing (CQ).

3.1 Priority Queueing (PQ)

Tento model funguje tak, že pakety v nejvyšší prioritní třídě mají přednost před pakety v ostatních prioritních třídách. Mohou získat 100% přenosové kapacity na úkor ostatních, může tedy dojít k takzvanému „vyhladovění“ toků (starvation) s nižší prioritou. PQ nachází použití v případech, kdy pakety s vysokou prioritou mají pouze minoritní zastoupení v přenášeném objemu dat.

Do vysoké priority řadíme toky náchylné na zpoždění paketů a jeho vysoký rozptyl, přiřazením vysoké priority snižujeme pravděpodobnost ztráty a zvyšujeme pravděpodobnost včasného odeslání.

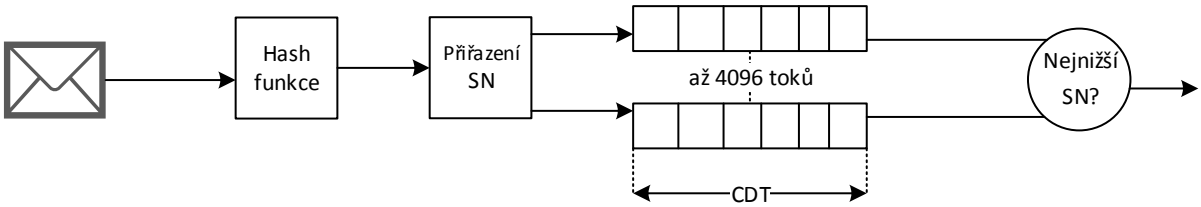
Nevýhodou je, že při obsazení paměti dochází k ztrátám paketů i v nejvyšší prioritní třídě.



Obr. 3.1 – Priority Queueing

3.2 Weighted-Fair Queueing (WFQ)

Hlavním rozdílem od metody Priority Queing je především upřednostnění menších paketů před velkými v případě zahlcení přenosového kanálu [5].



Obr. 3.2 – Princip WFQ

Klasifikace paketů probíhá rozčleněním paketů do toků hash funkcí, jejíž vstupními parametry jsou:

- zdrojová a cílová IP adresa,
- protokol, definovaný v záhlaví 3. vrstvy ISO/OSI modelu,
- zdrojový a cílový port.

Počet toků, do kterých se pakety klasifikují, může dosahovat 2^k, kde $k = 4,5,...12$. Ve WFQ je používáno sekvenční číslo (SN) odlišné od stejnojmenného čísla používaného v transportní vrstvě ISO/OSI modelu.

Ve WFQ je přiřazováno na třetí vrstvě a je popsáno vztahem:

$$SN_n = SN_{n-1} + W \times L \ , \tag{3.1}$$

kde SN_{n-1} je poslední přiřazené nebo odeslané sekvenční číslo, W je váha vypočtená podle vztahu

$$W = \frac{32384}{IP\_PRECEDENCE + 1} \tag{3.2}$$

a L je délka n-tého paketu v bajtech.

Počet paketů v každém paměti je omezen. Velikost fronty definuje maximálním množstvím paketů parametr Congestive Discard Threshold (CDT). Další parametr Hold-Queue Limit (HQL) udává maximální počet paketů uložených ve frontách všech toků.

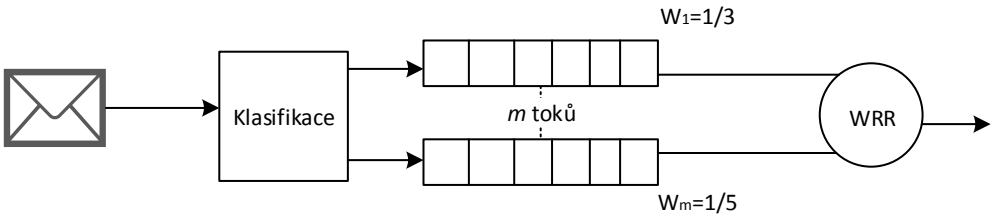
Paket náležící x-tému toku s již Pk_x uloženými pakety je přijat, pokud není překročen celkový dovolený počet paketů v paměti HQL a současně maximální dovolený počet paketů x-tého toku CDT

$$\begin{aligned} &\sum_{i}^{n} Pk_i \le HQL \\ &Pk_x \le CDT \end{aligned} \ . \tag{3.3}$$

Pokud paket nesplňuje podmínku s maximálním dovoleným počtem paketů, může být ještě uložen, jestliže v ostatních frontách existuje paket s větším sekvenčním číslem. Jestliže ano, zahodí se paket s větším sekvenčním číslem a místo něj je do fronty vložen paket číslem menším.

3.3 Custom Queueing (CQ)

V tomto přístupu jsou příchozí pakety klasifikovány do m pamětí a přenosová kapacita výstupního rozhraní je rozdělena na m dílů. Cisco konkrétně hovoří dle [5], že m je rovno 16. Jednotlivým pamětem se nastavuje váha, podle které je pak přiřazena přenosová kapacita. Součet všech přiřazených vah je 100%.



Obr. 3.3 – Princip CQ

Použitím CQ pouze rozdělíme dostupnou kapacitu v_p mezi zvolené toky bez priorit.

Při počtu bajtů odeslaných během jednoho cyklu QV_i je garantovaná rychlost i-tého toku C_i přibližně rovna

$$C_i \approx \frac{QV_i}{\sum_{i=1}^{m} QV_i} \times v_p \text{ [kbit/s].} \qquad (3.4)$$

4 Simulace

Na demonstraci činnosti algoritmů popsaných v předchozí kapitole byl vybrán simulační program OMNeT++ a framework INET[1].
Simulační scénáře byly zvoleny pro porovnání režimů front FIFO, PQ, WFQ a pro potvrzení předpokladů reakce na konkrétní druh síťového provozu.

4.1 Simulační program OMNeT++

Jedná se o objektově orientovaný simulátor diskrétních událostí napsaný v programovacím jazyku C++ využívající vývojové prostředí Eclipse [6]. Jeho použití je široké, od simulace drátových a bezdrátových sítí, protokolů až po simulaci hardwarových architektur.
OMNeT++ jako takový nesimuluje nic konkrétního, je potřeba na něj nahlížet spíše jako na prostředek k realizaci simulace. Skládá se z funkčních bloků tzv. modulů a ty se skládají dohromady podle vlastní potřeby.
Moduly lze mezi sebou propojit. K tomu slouží porty (gates). Dále zde funguje předávání zpráv mezi jednotlivými moduly, přičemž zprávy mohou přenášet libovolné datové struktury.
OMNeT++ je vydán pro Windows, MAC OS a Linux, což nahrává jednoduché přenositelnosti a je zdarma pro akademické a neziskové použití. Komerční alternativa se nazývá OMNEST.
V OMNeT++ se používá jazyk NED pomocí kterého lze popsat simulační strukturu. Dále slouží např. k deklaraci, spojování a skládání modulů.
NED jazyk má mnoho výhod jako například:

- hierarchičnost – způsob vyrovnání se s komplexností simulace a to rozdělením na několik menších a jednodušších podproblémů,
- dědičnost – zvyšuje použitelnost kódu, výhoda naprogramování v C++.

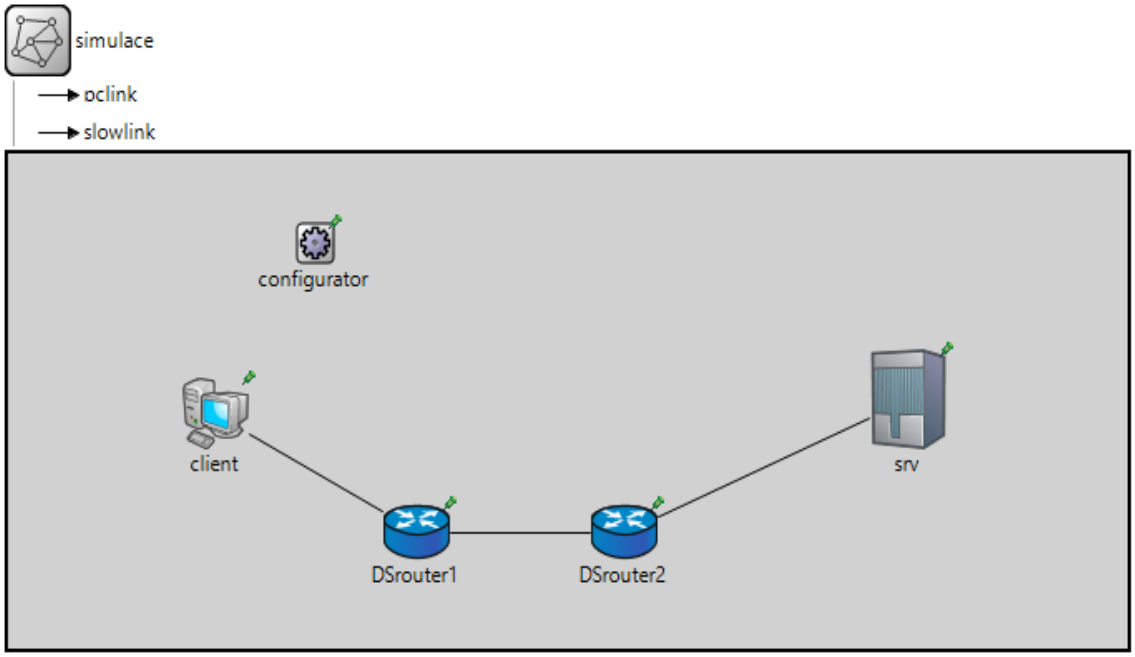
Další informace ohledně OMNeT++ jsou uvedeny v manuálu [7]. Framework INET rozšiřuje OMNeT++ o podporu síťových protokolů a informace a jeho možnosti jsou popsány v dokumentaci [8].

---

[1] Použité verze: OMNeT++ verze 4.6, INET verze 2.6.0, dostupné na http://www.omnetpp.org/ a http://inet.omnetpp.org/

4.2 Simulační parametry

Všechny simulační scénáře uvažují stejnou topologii, přičemž v jednotlivých scénářích budou měněny jednotlivé režimy fronty na rozhraní *ppp[1]* směrovače *DSrouter1*. Topologii lze vidět na obr. 4.1. Komunikace v simulačních scénářích pro jednoduchost probíhá jednostranně - od klienta k serveru. Od serveru ke klientovi probíhá pouze potvrzování o přijetí a komunikace spjatá s navazováním spojení u TCP protokolu.



Obr. 4.1 - Simulovaná topologie

Dále se uvažuje, že každý směrovač má jiného vlastníka, kteří mají mezi sebou sjednaný Service-level Agreement (SLA). Pro demonstraci simulačních scénářů je omezování toku vytvořeno nastavením pevnou šířkou přenosového pásma mezi *DSrouter1* a *DSrouter2*, linka je oboustranně omezována na 4 Mbit/s. Linka mezi směrovači a koncovými stanicemi má rychlost nastavenou na 100 Mbit/s. *Client* a *srv* jsou pouze ilustrativním příkladem, lze si pod nimi představit i více koncových stanic. Pro simulaci je ale podstatné, že se jedná o prvky generující známý síťový provoz.

4.3 Generované toky

Dalším společným parametrem je generovaný tok. OMNeT disponuje generátory toku na čtvrté vrstvě ISO/OSI modelu, využívány budou následující:

- *UDPBasicApp* – generuje zprávy o dané velikosti s nastaveným intervalem odesílání,
- *TCPSessionApp* – TCP přenos souboru o zvolené velikosti s nastaveným intervalem odesílání,
- *UDPBasicBurst* - posílá zprávy o dané velikosti ve shlucích s nastaveným intervalem odesílání a pauzy mezi odesíláním.

Tyto, v OMNeTu nazývané aplikace, mají za úkol simulovat VoIP hovor kodeku G.711, tok HD videa, tok webových stránek a aplikace běžící na pozadí.

Tab. 4.1 - Generovaný tok

Simulovaný tok	**Velikost zprávy [B]**	**Perioda odesílání [ms]**	**Přenosová rychlost na L2 [kbit/s]**
VoIP hovor	92	10	101,6
HD video	15015 (1500 na L3)	33,3	3612
aplikace	100	5	68,2[2]
webové stránky	celkem 10MB	exp(10ms)	-

V tab. 4.1 jsou konkrétní parametry nastavované v simulaci. První 3 toky jsou generovány UDP aplikacemi a mají pevnou periodu odesílání, zatímco poslední tok generovaný TCP aplikací je generován v nepravidelných intervalech s exponenciálním rozložením. V tabulce není uvedeno, že tok *aplikace* má nastavenou dobu aktivity na 600 ms a dobu neaktivity na 1300 ms. Paket toku *HD videa* je ihned po vygenerování rozdělen na pakety o velikosti 1500 bajtů a ty jsou odeslány na přenosové rozhraní.

[2] *Doba neaktivity již zohledněna*

Generovány jsou tedy 2 toky s pakety odesílanými ve shlucích – *aplikace a HD video*. Jeden tok s pravidelným odesíláním paketů – *VoIP hovor* a jeden s nepravidelným odesíláním paketů – *webové stránky.*

4.4 Scénář: First in First out (FIFO)

Tento scénář ukazuje chování směrovače při základním nastavení režimu fronty s omezenou velikostí 100 paketů. Při plné paměti ve směrovači dochází k zahazování paketů přicházející na vstupní rozhraní bez rozdílu (Drop-Tail).
V tomto scénáři je očekávána ztráta paketů bez ohledu na přiřazenou prioritu, vliv má pouze čas příchodu, zaplnění fronty a vytížení linky.
TCP tok je generován pouze jeden, ale při více TCP tocích by se projevila i absence aktivního řízení fronty (AQM) a došlo by ke vzniku globální TCP synchronizace, která by zapříčinila větší ztrátu paketů.

4.4.1 Simulace hodnot

Pro získání výsledků proběhla simulace 100x. Při každém spuštění probíhala s odlišným nastavením generátoru náhodných čísel, tato vlastnost se OMNeTu jmenuje jádro (seed). Je to jeden z mnoha parametrů generátorů náhodných čísel, má za úkol simulovat stochastické chování veličin v systému. Bylo ponecháno základní nastavení OMNeTu, kdy jádrem bude pořadové číslo simulačního úseku.
Pro získání ztracených paketů TCP aplikace bylo nutno přidat mezi generující TCP aplikaci a síťovou vrstvu modul *thruputMeter*. Zbylé veličiny nutné pro vyhodnocení simulace využívaly implementovanou statistiku v OMNeTu získanou pomocí tzv. Signals.
Pro skupinu toků (služeb) popsaných v kapitole 4.3 byly ze simulací získány následující hodnoty:

- průměrná velikost fronty dosahovala *79,87 ± 0,26* paketů,
- průměrné zdržení paketu ve frontě bylo (*146,69 ± 0,48)* ms,
- průměrně ztraceno *(1334,93 ± 14,53)* B,
- průměrná ztráta TCP toku byla *(18385,13 ± 332,18)* B.

Při detailnějším zkoumání jednotlivých toků lze výsledné hodnoty uspořádat do tabulky.

Tab. 4.2 - Výsledky simulací FIFO

Tok	Průměrné zdržení paketu (end to end) [ms]	Pravděpodobnost ztráty paketů [%]
VoIP hovor	*148,242 ± 0,51*	*1,82 ± 0,02*
HD video	*162,14 ± 0,54*	*15,09 ± 0,20*
aplikace	*137,24 ± 0,29*	*6,75 ± 0,06*
webové stránky	*-* [3]	*0,83 ± 0,02*

4.4.2 Vyhodnocení výsledků

Výsledky simulací ukazují, že paměť směrovače byla průměrně zaplněna ze 4/5. Průměrné zdržení paketu pro *VoIP* bylo na hranici přijatelnosti (150ms) [9]. Největší ztrátovost vykazoval tok *HD video*, jehož v průměru každý 6,6 paket byl zahozen.

4.5 Scénář: Priority Queuing

Tento scénář demonstruje snížení zpoždění při použití systému s prioritami. Především tokům zařazeným do vysokých prioritních tříd, naopak u aplikací zařazených v nejnižší prioritní třídě je očekáváno zvýšení odezvy než při režimu fronty FIFO. Režim fronty bude opět nastaven na směrovači *DSrouter1* u vysílacího rozhraní linky *DSrouter1 - DSrouter2*.

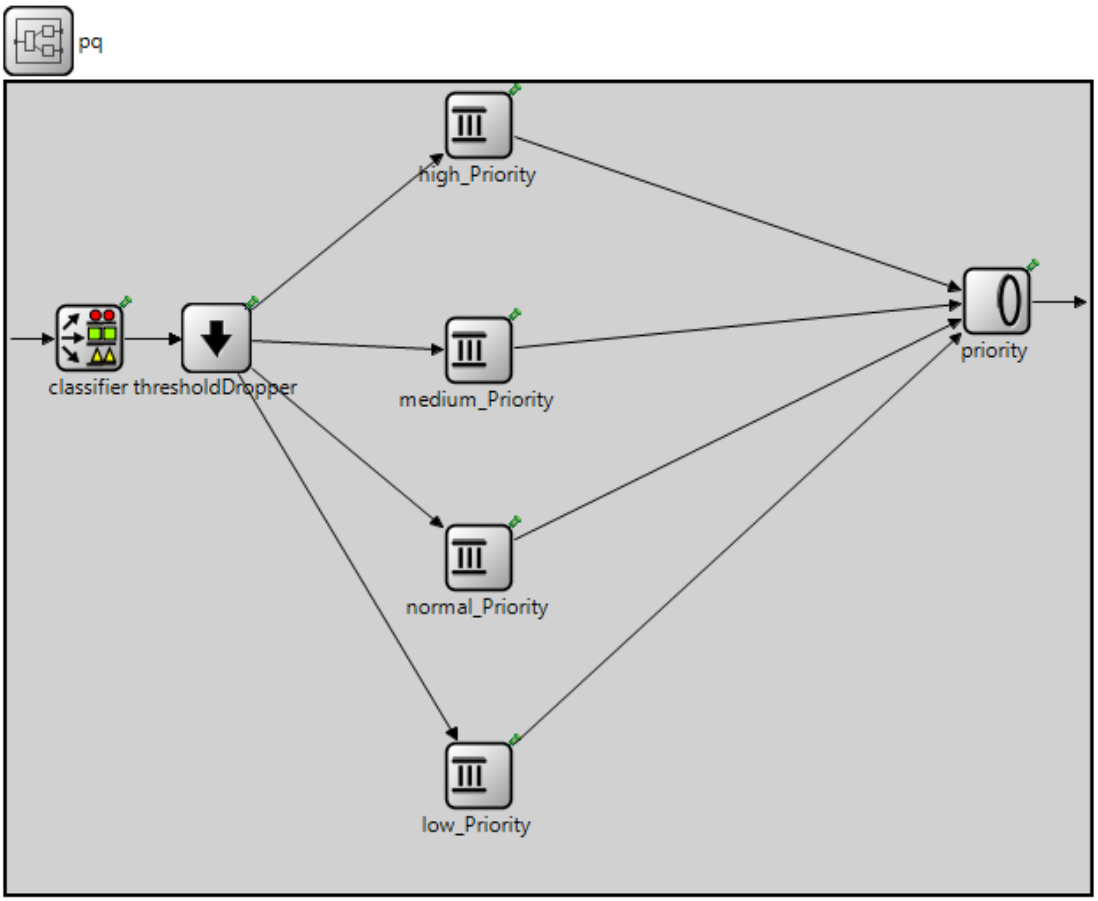
4.5.1 Sestavení PQ v OMNeT++

Pro sestavení PQ bylo možné využít moduly frameworku INET a z nich vytvořit vlastní složený modul. Bloky v OMNeTu mohou být buď jednoduché (simple module) nebo složené z jednoduchých bloků (compound module). Tím lze zjednodušit vytváření nových simulací využitím stávajících modulů.

---

[3] Hodnoty nebyly zaznamenány, protože OMNeT vyhodnotil pouze spuštění přenosu jako jednu událost – odeslání 10MB

Využívala se i dědičnost OMNeTu, složený modul zdědil vlastnosti *IOutputQueue*, aby byla možnost nahradit základní režim fronty na směrovači FIFO za vlastní modul PQ.



Obr. 4.2 - PQ NED struktura

Dle obr. 4.2 klasifikaci paketů v OMNeT++ provádí modul pojmenovaný *classifier*, který příchozí tok do PQ roztřídí na základě zvolených DSCP značek do 4 prioritních tříd (tab. 4.3). Velikost jednotlivých front je nastavena na základní hodnoty 20, 40, 60, 80 paketů od nejvyšší priority po nejnižší [10]. Na skladbu front je využit modul *DroptailQueue*.

Před vstupem paketů do front je omezována jejich velikost pomocí modulu *ThresholdDropper*, který omezuje celkovou velikost všech front na hodnotu nastavenou v inicializačním souboru simulace.

Dále si lze povšimnout modulu plánovače *PriorityScheduler* pojmenovaný *priority*, který má při zavolání metody *requestPacket()* na starost výběr paketu pro odeslání na přenosové rozhraní. Paket z nižší prioritní třídy je vybrán pouze za

předpokladu, že v pamětech nadřazených front nejsou přítomny žádné pakety. V simulaci jsou pro klasifikaci nastaveny následující parametry.

Tab. 4.3 - Klasifikace v PQ

Prioritní třída	**DSCP pole (vyjádřeno dekadicky)**	**Odpovídá generovanému toku**
Nejvyšší priorita	34	VoIP hovor
Střední priorita	26	HD video
Normální priorita	18	-
Nejnižší priorita	Nezařazený tok	aplikace, webové stránky

4.5.2 Simulace hodnot

Simulováno bylo opět 100 úseků s délkou 60 sekund. Získány byly následující hodnoty:

- průměrná velikost fronty v nejvyšší prioritě činila *0,50 paketu,*
- průměrná velikost fronty ve střední prioritě činila *4,90 paketů*,
- průměrná velikost fronty v nízké prioritě činila *44,43 ± 0,05 paketů*,
- průměrná ztráta TCP toku byla (*13216,03 ± 215,12)* B.

Na následující tabulce lze vidět výsledné hodnoty simulací.

Tab. 4.4 - Výsledky simulací PQ

Tok	**Průměrné zdržení paketu (end to end) [ms]**	**Pravděpodobnost ztráty paketů [%]**
VoIP hovor	*1,76*	*0*
HD video	*32,04*	*0*
aplikace	*476,56 ± 0,78*	*18,06 ± 0,1*
webové stránky	-	*1,13 ± 0,02*

4.5.3 Vyhodnocení výsledků

U toku s *HD videem* došlo ke ztrátě jednoho paketu. Detailnější analýza výsledků simulace odhalila, že se nejedná o ztrátu, nýbrž o probíhající přenos posledního paketu. Nastavena je pevná doba ukončení simulace, která přenos přeruší a dojde k výpočtu ztráty paketů jako rozdíl odeslaných a přijatých paketů. U tohoto scénáře lze při ztrátě jednoho paketu tvrdit, že nedošlo k žádné ztrátě. U ostatních scénářů je tato chyba statisticky nevýznamná.

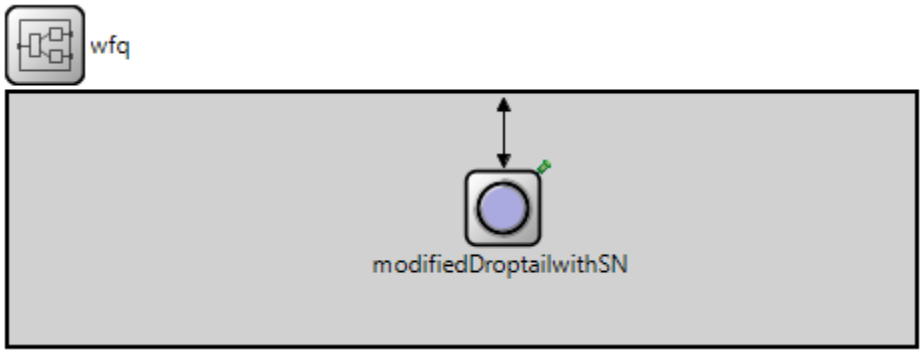
Nejvíce postiženým byl generovaný tok *aplikace*, který spolu s tokem *webových stránek* byl v nejnižší prioritní třídě.

4.6 Scénář: Weighted-Fair Queueing

Framework INET nedisponuje plánovačem WFQ, proto byly navrhnuty moduly realizující plánovač WFQ. Scénář byl především navržen k porovnání s ostatními scénáři a pro ověření správné implementace. Tedy tak, jak by se choval na reálném směrovači.

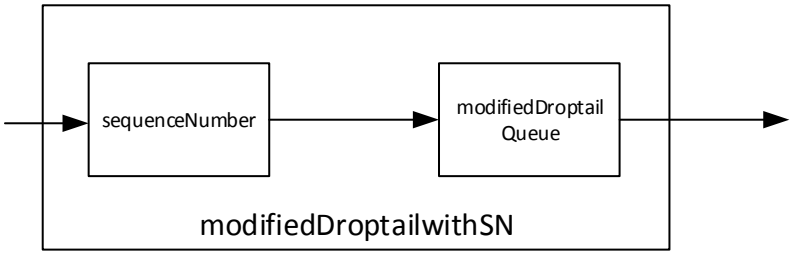
Moduly bylo nutno pro použití plánovače doprogramovat. Strukturu pomocí jazyka NED, funkčnost pomocí jazyka C++.

Struktura modulů byla vytvořena tak, že po příchodu paketu do WFQ nejprve prochází složeným modulem *modifiedDroptailwithSN*.



Obr. 4.3 - Struktura WFQ

Jedná se o modul, který je složen z dvou dalších modulů – *sequenceNumber* a *modifiedDroptail*. Ty tvoří jádro celého WFQ postaveného v OMNeTu. Průchod paketu moduly WFQ lze naznačit následovně.



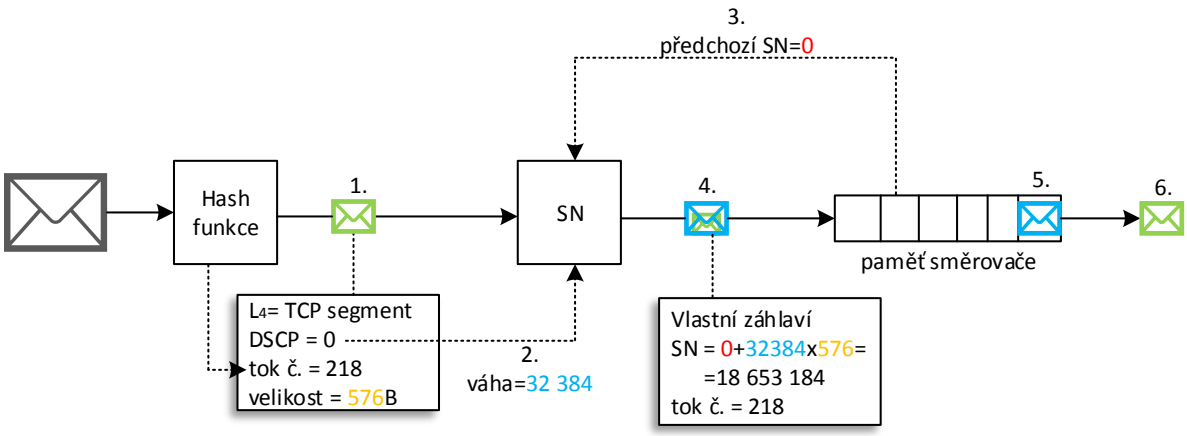
Obr. 4.4 - Struktura modulů

V klíčovém modulu *sequenceNumber* dochází k rozčlenění toků. To ve WFQ probíhá na základě známých parametrů (viz 3.2). Tyto parametry přijme hash funkce a výstupem z hash funkce je číslo modulo maximálním počtem toků, které dosahuje v mocninách 2 hodnot od 16 do 4096. Jedná se tedy o identifikátor toku. Tokem v tomto případě rozumíme přenos od zařízení k druhému se stejnými vstupními parametry dané hash funkce. Pokud dojde ke změně některého z parametrů, dochází také ke změně identifikátoru toku, avšak pro jeden identifikátor toku existuje více kombinací vstupních parametrů.
Po výpočtu identifikátoru toku je paketu přiřazeno sekvenční číslo. Jedná se o číslo závislé na předchozí hodnotě. Při přiřazování sekvenčního čísla pro paket v novém toku se bere jako předchozí sekvenční číslo hodnota posledního paketu odešlého na výstupní rozhraní (viz kapitola 3.2).

Dále je sekvenční číslo, identifikátor toku a vstupní parametry zapouzdřeny do vlastního záhlaví a zaslány modulu *modifiedDroptailQueue* pro další zpracování. Jak již název napovídá, jedná se o mechanismus Drop-Tail modifikovaný pro použití u WFQ. Již nelze hovořit o režimu fronty FIFO, pakety jsou řazeny dle svého sekvenčního čísla - od nejnižšího po nejvyšší. U WFQ se jedná o velmi důležitý aspekt, protože plánovač má za úkol vybrat pro odeslání paket s nejnižším sekvenčním číslem ze všech front. Při seřazené frontě lze využít pouze volání metody *requestPacket()*, která vybere první paket ve frontě.
Aby plánovač nemusel procházet všechny fronty (až 4096), lze se stejnou funkčností použít pouze frontu jednu. Díky tomuto kroku již nebude potřeba sledovat konfigurovatelný parametr Hold-Queue Limit. Ten udává maximální počet paketů ve všech frontách, lze ho tedy použít jako maximální velikost fronty. Při přesáhnutí HQL jsou pakety zahozeny.

4.6.1 Příklad komunikace

Tato kapitola se bude zabývat ukázkou principu činnosti funkčních bloků sestavené simulace v OMNeTu a bude používat konkrétní čísla zaznamenaná z běhu simulačního scénáře.



Obr. 4.5 - Příklad průchodu paketu

Na obr. 4.5 je možno vidět průchod IP paketu o velikosti 576 bajtů. V prvním bodu už paket prošel hash funkcí a má přidělený identifikátor toku. Generovanému toku TCP aplikace (*webové stránky*) je ve scénáři přidělován identifikátor toku číslo 218.

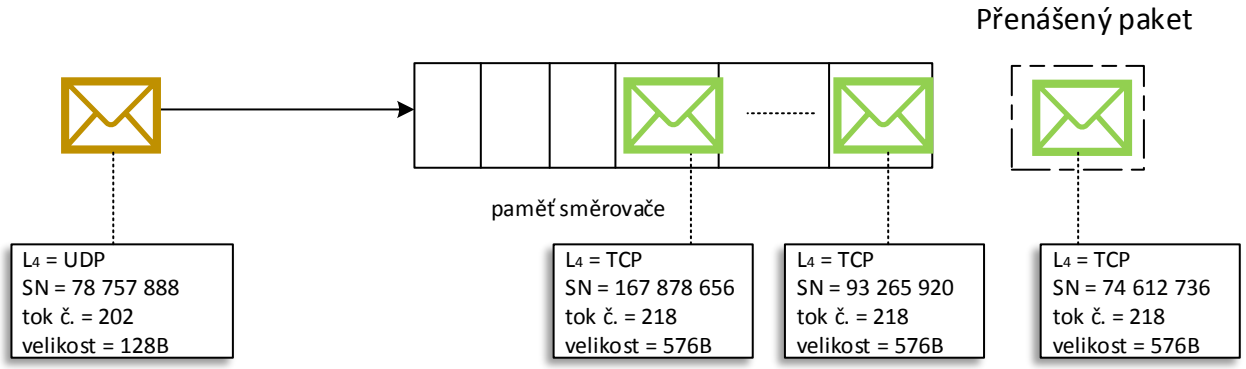
V druhém bodě se paket nachází v bloku SN, kde se vypočte z DSCP značky váha (viz vztah (3.2)). Pro přidělení sekvenčního čísla jsou potřeba 3 hodnoty: váha, velikost a předchozí sekvenční číslo. V třetím bodě již zbývá pouze zjistit poslední vyjmenovanou hodnotu. V tomto konkrétním případě je paměť směrovače prázdná, proto je vrácena nulová hodnota. Pokud by byl v paměti pozdržen alespoň jeden paket, bylo by předchozí sekvenční číslo nenulová hodnota – buď posledního odeslaného, nebo pozdrženého paketu. Po získání předchozího sekvenčního čísla dochází v bloku SN k výpočtu čísla sekvenčního (viz vztah (3.1)).

Ve čtvrtém bodě obr. 4.5 je využíváno vlastní záhlaví pojmenované *SNmessage*. To slouží hlavně pro přenos sekvenčního čísla a identifikátoru toku[4] mezi moduly *sequenceNumber* (hash funkce a SN) a *modifiedDroptailQueue* (paměť směrovače). Před pozdržením paketu v paměti směrovače jsou otestovány podmínky (3.3), pokud jsou splněny, paket je pozdržen v paměti.

[4] *ve skutečnosti ještě přenáší IP adresy, velikost, protokol a porty pro možnost kontroly v grafickém módu*

V šestém bodě opouští paket paměť směrovače.

Na následujících obrázcích bude ukázána práce se sekvenčními čísly uvnitř paměti směrovače. Všechny pakety přicházejí a jsou uloženy s navrženým záhlavím *SNmessage*, ze kterého jsou pro stručnost vypsány jen sekvenční čísla, identifikátory toku a velikosti.



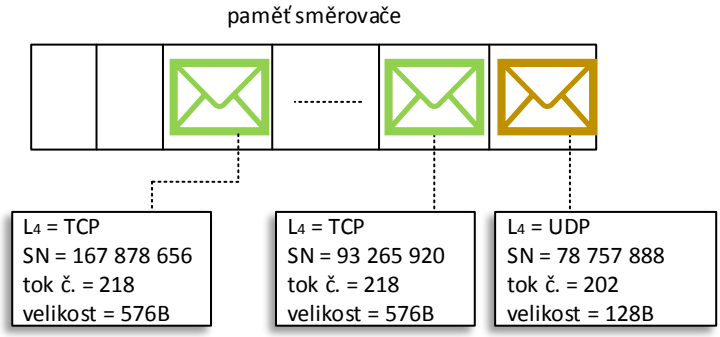
Obr. 4.6 - Příklad přicházejících toků

Na obr. 4.6 přichází paket toku simulující aplikace do paměti směrovače s dvěma pozdrženými pakety. Doby příchodu paketů jsou vidět v tabulce tab. 4.5.

Tab. 4.5 - Doba příchodu paketů

Sekvenční číslo	Příchod do paměti [ms]
74 612 736	12,41
93 265 920	12,46
167 878 656	14,79
78 757 888	15

Do paměti směrovače přicházejí zelené pakety toku *webových stránek*. Paměť je postupně zaplňována, protože na přenosovém rozhraní dochází k přenosu paketu se sekvenčním číslem *74 612 736.* V době 15ms přichází hnědý paket toku *aplikace*, jehož předchozí sekvenční číslo je vzato číslo právě přenášeného paketu. Jak zelené, tak i hnědé pakety mají DSCP značku nulovou. Pokud porovnáme hodnoty sekvenčních čísel zelených paketů s hnědými, tak bude vidět znatelný rozdíl ve velikosti sekvenčních čísel. Příčinou je rozdíl ve velikosti paketů.



Obr. 4.7 - Příklad řazení paměti směrovače

Na dalším obrázku (obr. 4.7) už byl paket toku *aplikací* přesunut do paměti. Při přesouvání došlo k porovnání sekvenčních čísel algoritmem, aby prvním odchozím paketem byl paket s nejnižším sekvenčním číslem.

Seřazení je také vidět při porovnání obr. 4.7 a tab. 4.5.

4.6.2 Simulace hodnot

Stejně jako v minulých scénářích simulace proběhla 100x. Nasimulovány byly následující hodnoty:

- průměrná velikost fronty dosahovala *56,85 ± 0,51* paketů,
- průměrné zdržení paketu v paměti směrovače bylo *(96,30 ± 0,77)* ms,
- průměrně ztraceno *(299,22 ± 16,65)* paketů,
- průměrná ztráta TCP toku byla *(14676,41 ± 887,85)*B.

Výsledné hodnoty generovaných toků lze uspořádat do tab. 4.6.

Tab. 4.6 - Výsledky simulací WFQ

Tok	**Průměrné zdržení paketu (end to end) [ms]**	**Pravděpodobnost ztráty paketů [%]**
VoIP hovor	*50,69 ± 1,57*	*0,55 ± 0,03*
HD video	*293,30 ± 7,60*	*3,80 ± 0,18*
aplikace	*69,16 ± 1,86*	*1,05 ± 0,05*
webové stránky	-	*1,02 ± 0,05*

4.6.3 Vyhodnocení výsledků

Nejprve se podle vztahu (3.2) (ze znalosti IP precedence simulovaných paketů) vypočítají jednotlivé váhy generovaných toků:

$$\begin{aligned} W_{VoIP} &= \frac{32384}{4+1} = 6476, \\ W_{HDvideo} &= \frac{32384}{3+1} = 8096, \\ W_{aplikace,webové\_stránky} &= \frac{32384}{0+1} = 32384. \end{aligned} \tag{4.1}$$

Pokud se označí přírůstky sekvenčních čísel jako ΔSN, platí:

$$\Delta SN = W \times L \ . \tag{4.2}$$

Ze známých velikostí generovaných zpráv, kde L je délka paketu (tab. 4.1) lze dále dopočítat:

$$\begin{aligned} \Delta SN_{VoIP} &= 6476 \times (92+8+20) = 777120, \\ \Delta SN_{HDvideo} &= 8096 \times (1472+8+20) = 12144000, \\ \Delta SN_{aplikace} &= 32384 \times (100+8+20) = 4145152, \\ \Delta SN_{webové\_stránky} &= 32384 \times 580 = 18782720 \end{aligned} \tag{4.3}$$

U toku *webové stránky* je velikost paketů proměnná, proto je brána průměrná velikost paketu. Přírůstek *VoIP toku* je nejmenší ze všech generovaných toků. Vypočtené hodnoty ze vztahu (4.3) lze vztáhnout k přírůstku ΔSN_{VoIP} a zjistit kolikrát větší jsou přírůstky ostatních toků:

$$\begin{aligned} \frac{\Delta SN_{HDvideo}}{\Delta SN_{VoIP}} &= 15,63; \\ \frac{\Delta SN_{aplikace}}{\Delta SN_{VoIP}} &= 5,33; \\ \frac{\Delta SN_{webové\_stránky}}{\Delta SN_{VoIP}} &= 24,16. \end{aligned} \tag{4.4}$$

Pro intenzitu odesílání paketů λ platí, že je převrácenou hodnotou periody odesílání t_p:

$$\lambda = \frac{1}{t_p} \ [\mathrm{s}^{-1}]. \tag{4.5}$$

Pokud opět dojde ke vztažení k toku *VoIP hovoru*:

$$\begin{aligned}
\frac{\lambda_{HDvideo}}{\lambda_{VoIP}} &= \frac{t_{p_VoIP}}{t_{p_HDvideo}} = \frac{0,010}{0,0033} = 3; \\
\frac{\lambda_{aplikace}}{\lambda_{VoIP}} &= \frac{t_{p_VoIP}}{t_{p_aplikace}} = \frac{0,010}{0,005} = 2; \\
\frac{\lambda_{webové_stránky}}{\lambda_{VoIP}} &= \frac{t_{p_VoIP}}{t_{p_webové_stránky}} = \frac{0,0100}{0,0055} = 1,8.
\end{aligned} \tag{4.6}$$

Z výše zmíněných výpočtů a tab. 4.6 lze konstatovat, že nejmenší sekvenční čísla jsou přidělována *VoIP* toku, proto tento tok dosahuje nejmenšího průměrného zdržení a pravděpodobnosti ztráty. Druhých nejlepších hodnot průměrného zdržení a pravděpodobnosti ztráty dosahuje tok *aplikace*, ačkoliv jsou tomuto toku přidělovány přibližně 5,5 krát horší sekvenční čísla než *VoIP hovoru*. Téměř dvojnásobná intenzita odesílání paketů toku *aplikace* $\lambda_{aplikace}$ oproti intenzitě *VoIP* toku λ_{VoIP} znamená, že v době aktivního režimu toku *aplikace* nemusejí být pakety obsluhovány a jejich počet v paměti směrovače může být vyšší než u paketů z toku *VoIP*.

Zbývají už jen tok *webové stránky* a tok *HD videa*. U toku *webové stránky* je pravděpodobnost ztráty paketu menší než u toku *HD videa*, i když jeho přírůstek sekvenčních čísel je mnohem větší než u toku *HD videa*. Je to způsobeno použitým protokolem čtvrté vrstvy ISO/OSI modelu, tj. TCP protokol.

Závěr

Cílem této bakalářské práce bylo vytvoření simulačních scénářů a vytvoření funkčního modelu Weighted Fair Queueing.

V rešeršní části bakalářské práce došlo k prostudování v praxi používaných metod omezování a tvarování toku ve směrovačích. V praktické části jsem se seznámil se simulačním programem OMNeT++, ve kterém jsem také vytvořil simulační scénáře. Ve scénářích byl generován referenční datový tok. Tento tok se skládal z třech toků UDP, které simulovaly tok *VoIP hovoru, HD videa, aplikací* s pevně nastavenou dobou odesílání a jednoho TCP toku, simulujícího tok *webových stránek*, s proměnnou dobou odesílání.

Pokud se zaměříme na generovaný tok *VoIP hovoru*, tak můžeme konstatovat, že z hlediska ztrátovosti a zdržení paketu bylo dosaženo nejlepších hodnot ve scénáři Priority Queueing následovaný scénářem Weighted Fair Queueing. Dalším zajímavým tokem byl tok *HD videa*, který byl specifický především pro svou nejvyšší velikost generované zprávy ze všech UDP toků. Tok *HD videa* dosahoval nejlepších hodnot ve scénáři Priority Queueing, stejně jako již výše zmiňovaný *VoIP tok*. V dalších scénářích záleželo na tom, jaké parametry u *HD videa* sledujeme: ztrátovost paketů *HD videa* byla výrazně nižší ve WFQ, ale průměrná doba zdržení paketu téměř dvojnásobná oproti FIFO.

Ve výsledcích došlo k potvrzení předpokladů, že v PQ dochází k nejmenšímu průměrnému zdržení u paketu zařazených ve vyšších prioritních třídách než v třídách nižších. To samé platí i u pravděpodobnosti ztráty paketů na úkor toku *aplikace*. U režimu fronty FIFO záleželo pouze na času příchodu paketu a zaplnění paměti směrovače. U scénáře WFQ dosahovaly pakety menších toků lepších hodnot nežli velkých.

Funkční model WFQ byl vytvořen tak, že jádro modelu bylo vytvořeno v C++ a struktura byla rozdělena mezi dva moduly. Tyto moduly mezi sebou komunikovaly pomocí vlastního záhlaví, které přenášelo sekvenční číslo, identifikátor toku a pro kontrolní účely i zdrojovou, cílovou IP adresu, zdrojový, cílový port a protokol čtvrté vrstvy ISO/OSI modelu.

V budoucnu bych se chtěl zabývat úpravou vytvořeného modelu WFQ a dosáhnout zařazení tohoto modelu do frameworku INET.

Seznam použitých zdrojů

[1] ALMQUIST, P. Type of Service in the Internet Protocol Suite: RFC: 1349. In: *Request for Comments* [online]. 1992 [cit. 2015-05-01]. Dostupné z: https://tools.ietf.org/html/rfc1349

[2] The Addition of Explicit Congestion Notification (ECN) to IP: RFC: 3168. In: RAMAKRISHNAN, K., TERAOPTIC NETWORKS, S. FLOYD, ACIRI, D. BLACK a EMC. *Request for Comments* [online]. 2001 [cit. 2015-05-01]. Dostupné z: https://tools.ietf.org/html/rfc3168

[3] PARK, Kun Il. 2005. *QOS in packet networks*. New York: Springer Science Business Media, xii, 243 p. ISBN 03-872-3389-X.

[4] J. HEINANEN, a R. GUERIN. 1999. A Single Rate Three Color Marker. *Request for Comments* [online]. [cit. 2015-04-27]. Dostupné z: https://tools.ietf.org/html/rfc2697

[5] ODOM, Wendell a Michael J CAVANAUGH. 2004. *Cisco DQOS exam certification guide: IP telephony self-study*. Indianapolis, IN: Cisco Press, xxxvi, 900 p. ISBN 15-872-0058-9

[6] *Eclipse* [online]. 2013 [cit. 2015-05-17]. Dostupné z: https://eclipse.org/

[7] VARGA, András. 2014. *OMNeT++ User Manual* [online]. 4.6. Dostupné z: www.omnetpp.org/doc/omnetpp/Manual.pdf

[8] *INET Framework for OMNeT++: Manual* [online]. 2012. [cit. 2015-04-27]. Dostupné z: http://omnetpp.org/doc/inet/api-current/inet-manual-draft.pdf

[9] *ITU-T G.114: One-way transmission time*. Geneva: TELECOMMUNICATION STANDARDIZATION SECTOR OF ITU, 2003. Dostupné také z: http://www.itu.int/rec/dologin_pub.asp?lang=e&id=T-REC-G.114-200305-I!!PDF-E&type=items

[10] *Cisco IOS Quality of Service Solutions Configuration Guide: Configuring Priority Queueing* [online]. 2005 [cit. 2015-05-10]. Dostupné také z: http://www.cisco.com/c/en/us/td/docs/ios/12_2/qos/configuration/guide/fqos_c/qcfpq.pdf

Seznam použitých obrázků a tabulek