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

POKROČILÉ MOŽNOSTI TVAROVÁNÍ DATOVÉHO TOKU V OS LINUX PRO SÍTĚ 802.3 A 802.11

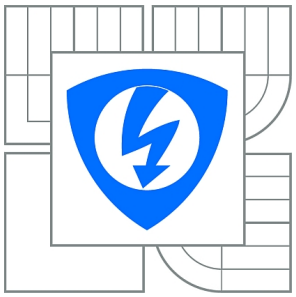
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE MICHAL PÁNEK
AUTHOR

BRNO 2013

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

POKROČILÉ MOŽNOSTI TVAROVÁNÍ DATOVÉHO TOKU V OS LINUX PRO SÍTĚ 802.3 A 802.11

ADVANCED FEATURES OF TRAFFIC SHAPING FOR 802.3 AND 802.11 NETWORKS UNDER OS LINUX

BAKALÁŘSKÁ PRÁCE

BACHELOR'S THESIS

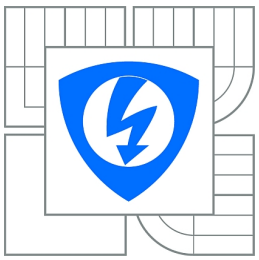
AUTOR PRÁCE — MICHAL PÁNEK

AUTHOR

VEDOUCÍ PRÁCE — Ing. JURAJ SZŐCS

SUPERVISOR

BRNO 2013



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Bakalářská práce

bakalářský studijní obor
Teleinformatika

Student:	Michal Pánek	***ID:***	136570
Ročník:	3	***Akademický rok:***	2012/2013

NÁZEV TÉMATU:

Pokročilé možnosti tvarování datového toku v OS Linux pro sítě 802.3 a 802.11

POKYNY PRO VYPRACOVÁNÍ:

Prozkoumejte možnosti, které OS Linux nabízí v tvarování datového toku. Definujte jednotlivé nástroje, které jsou nezbytně nutné pro tvarování provozu. Sestrojte stručný přehled metod, které lze v OS Linux použít pro tvarování, popište jejich chováni v sítích 802.3 a 802.11 a zjistěte jejich výhody/nevýhody. Vyberte si nejvhodnější metody, které byste použili pro tvarování provozu v sítích 802.3 a 802.11, popište je a zdůvodněte své rozhodnuti. Vytvořte si jednoduchou testovací bezdrátovou a drátovou síť, kde si zvolené metody odzkoušíte v praktické situaci. Nastavte si vhodné limity pro testovací účely a měřením ověřte funkčnost zvolených metod, výsledky vyneste do přehledných grafů.

DOPORUČENÁ LITERATURA:

[1] GHEORGHE, L. Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter. Packt Publishing, 2006. 288 s. ISBN: 1-904811-65-5.
[2] XIAO, X. Technical, Commercial and Regulatory Challenges of QoS: An Internet Service Model Perspective. Elsevier Inc., 2009. 336 s. ISBN: 978-0-12-373693-2.

Termín zadání:	11.2.2013	***Termín odevzdání:***	5.6.2013

Vedoucí práce: Ing. Juraj Szőcs
Konzultanti bakalářské práce:

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

UPOZORNĚNÍ:

Autor bakalářské práce nesmí při vytváření bakalářské práce porušit autorská práva třetích osob, zejména nesmí zasahovat nedovoleným způsobem do cizích autorských práv osobnostních a musí si být plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení části druhé, hlavy VI. díl 4 Trestního zákoníku č.40/2009 Sb.

ABSTRAKT

Táto bakalárska práca sa zaoberá možnosťami tvarovania a kontroly dátového toku, v operačnom systéme Linux. Prvá časť práce sa zaoberá preskúmaním jednotlivých nástrojov potrebných pre prácu s dátovým tokom. Druhá časť práce sa zaoberá metodamy určenými na tvarovanie dátového toku. Z tých boli vybrané a popísané metody, určené pre použitie v štandartoch 802.3 a 802.11. Práca sa v druhej časti zamerala na metody Hierarchical Token Bucket a Class-based queueing. Tretia časť práce je praktická a zameraná aplikovaním metod na hardvér, ich meraním na jednotlivých štandardoch a spracovaním do grafoch.

KĽÚČOVÉ SLOVÁ

mangľovanie, tvarovanie, kontrola dátového toku, ipchains, iptables, netfilter, iproute2, qdics, pfifo, SFQ, TBF, HTB, CBQ

ABSTRACT

This bachelor work deals with possibilities of traffic shaping and control in OS Linux. First part of the work examines indiviual tools needed for working with data stream. The second part consideres methods intended for traffic shaping. From these methods intended for use in standarts 802.3 and 802.11 were selected and described. The second part of paper focused on Hierarchical Token Bucket and Class-based queueing method. The third part is the practical application of methods on the hardware, the measurement of the individual standards and processing into charts.

KEYWORDS

mangling, shaping, traffic control, ipchains, iptables, netfilter, iproute2, qdics, pfifo, SFQ, TBF, HTB, CBQ

PÁNEK, Michal *Pokročilé možnosti tvarování datového toku v OS Linux pro sítě 802.3 a 802.11*: semestrálny projekt. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2013. 54 s. Vedúci práce bol Ing. Juraj Szőcs

PREHLÁSENIE

Prehlasujem, že som svoj semestrálny projekt na tému „Pokročilé možnosti tvarování datového toku v OS Linux pro sítě 802.3 a 802.11“ vypracoval samostatne pod vedením vedúceho semestrálneho projektu, využitím odbornej literatúry a ďalších informačných zdrojov, ktoré sú všetky citované v práci a uvedené v zozname literatúry na konci práce.

Ako autor uvedeného semestrálneho projektu ďalej prehlasujem, že v súvislosti s vytvorením tohoto semestrálneho projektu som neporušil autorské práva tretích osôb, najmä som nezasiahol nedovoleným spôsobom do cudzích autorských práv osobnostných a/nebo majetkových a som si plne vedomý následkov porušenia ustanovenia § 11 a nasledujúcich autorského zákona č. 121/2000 Sb., o právu autorském, o právoch súvisejúcich s právom autorským a o zmeně niektorých zákonov (autorský zákon), vo znení neskorších predpisov, vrátane možných trestnoprávnych dôsledkov vyplývajúcich z ustanovenia časti druhé, hlavy VI. diel 4 Trestného zákoníka č. 40/2009 Sb.

Brno ....

(podpis autora)

POĎAKOVANIE

Rád bych poděkoval vedoucímu diplomové práce panu Ing. Jurajovi Szőcsovi, za odborné vedení, konzultace, trpělivost a podnětné návrhy k práci.

Brno ....

(podpis autora)

OBSAH

ZOZNAM OBRÁZKOV

ZOZNAM TABULIEK

ÚVOD

Linux je open source projekt, v posledných rokoch sa rozšíril a je používaný veľkou škálou uživateľov, administrátorov a zariadení.

V dnešnej dobe je niekoľko linuxových distribúcií ako sú napríklad Ubuntu, Fedora, SuSe, Debian a mnoho ďalších. Linux je veľmi rozmanitý a ako operačný systém priniesol veľa možností. Základnou riadiacou jednotkou systému je kernel. Pri správnej konfigurácii kernelu sa môže stať z osobného počitača firewall, server alebo veľmi vykonný router. Hlavnú bránu otvoril kontrole a modelovaniu dátového toku. Použitím správnych nástrojov, je možné kernelom filtrovať dátový tok veľmi dôkladne, na troch vrstvách TCP/IP modelu ako aj používania NAT a priameho zasahovania do hlavičiek paketov. Kernal taktiež dokáže tvarovať dátový tok podľa našich zadaných špecifikácií, použitím metod určených na spracovanie paketov. Metody používané kernelom sa správajú k paketom rozdielne. Jednotlivé metody je možné pomocou iných metod zlučovať do celkov, a vytvárať tak samostatné štruktrúry.

Požiadavky na kvalitu služieb stále rastú. Aby sa dalo vyhovieť týmto požiadavkám je potrebné mať dostačujúcu šírku pásma a tu dokázať správne rozdeliť medzi uživateľov a ich žiadané služby. Úlohou tejto práce je zaoberať sa nástrojmi na tvarovanie dátového toku a metodami použiteľnými v Linuxe, ako aj ich výber pre použitie v šandartoch 802.3 a 802.11.

1 NÁSTROJE PRE SPRÁVU A TVAROVANIE DÁTOVÉHO TOKU

Na správu a riadenie dátoveho toku sa používajú nástroje vytvorené pre OS Linux. Existuje celá škála nástrojov, ktoré dokážu spracovať dátový tok rôzne. Najčastejšími používanými funkciami nástrojov je filtrovanie, mangľovanie a tvarovanie dátového toku.

1.1 Ipchains

Ipchains je architektúra navrhnutá pre Linux kernel 2.0, 2.1 a 2.2, ktorá dokáže vymazať alebo zadávať nové pravidlá do filtrovaciej sekcie kernelu. Jednotlivé pravidlá, podľa ktorých sa vyhodnocujú pakety sú zadávané cez terminál, ten tvorí prístup k ipchains.

Aby bolo možné používať ipchains, je predom potrebné v linuxe nastaviť správnu nakonfiguráciu a povoliť root.
Konfigurácia ipchains pre kernel 2.0 [4]:

```
CONFIG_EXPERIMENTAL=y
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
CONFIG_IP_FIREWALL_CHAINS=y
```

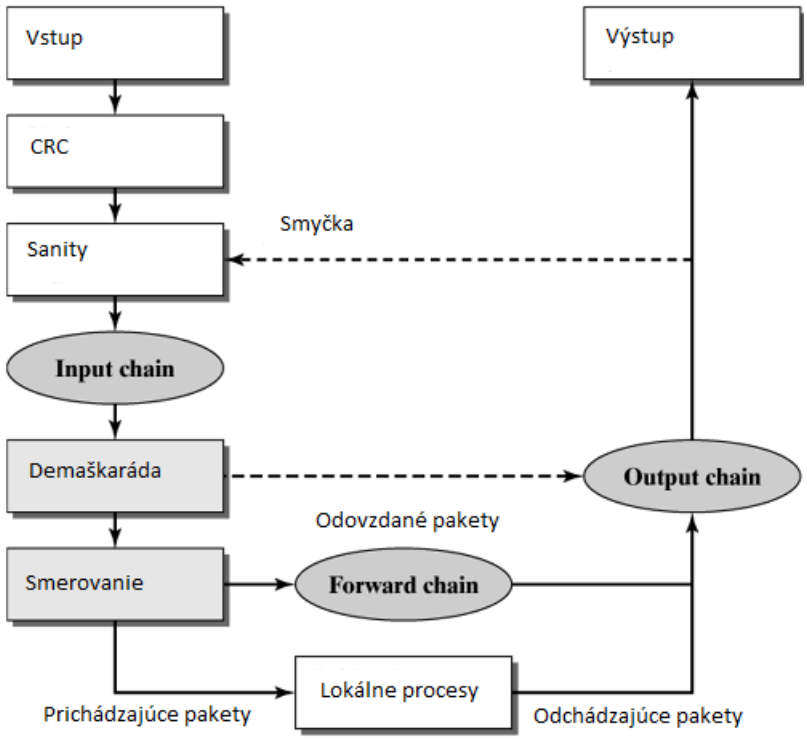
Konfigurácia ipchains pre kernel 2.1 a 2.2 [4]:

```
CONFIG_FIREWALL=y
CONFIG_IP_FIREWALL=y
```

Kernel na pakety aplikuje viaceré pravidlá, tie potom podľa svojho umiestnenia a funkcie tvoria reťaze(chains). V ipchains sú tri predom zadefinované reťaze:

- **Input chain** - vstupné pakety, pakety ktoré sú určené pre lokálnu sieť.
- **Forward chain** - preposlané pakety, pakety ktoré sú určené mimo lokálnu sieť alebo pre inú stanicu v lokálnej sieti na základe smerovania.
- **Output chain** - vychádzajúce pakety, pakety vytvorené Linuxom [1].

1.1.1 Architektúra ipchains



Obr. 1.1: Architektúra ipchains [1]

V obr.1.1 môžeme vidieť architektúru ipchains, podľa ktorej sa riadi kernel pri spracovaní paketov. Kernel spracuje pakety následovne:

1. **CRC:** Paket vchádza do linuxovej mašiny, spraví sa CRC kontrola, čím sa zistí bezúhonnosť paketu. Pokiaľ bol paket poškodený bude zamietnutý.
2. **Sanity:** Nasleduje ďalšie overovanie paketu nazývané Sanity. Sanity je znovu overovanie paketu proti poškodeniu, môže sa stať, že paket dokáže oklamať CRC aby tento poškodený paket nevnikol k jednotlivým reťazam, sanity ich nájde a odmietne pustiť ďalej.
3. **Input chain:** Pokiaľ paket prešiel skrz CRC a sanity, dostáva sa k input. Tu sa rozhoduje čo bude s paketom. Môže nastať jedna z troch možností ACCEPT - pri ktorej je paket pustený ďalej, DENY - paket je zahodený, REJECT-paket je odmietnutý a späťne odosiela správu pomocou protokolu ICMP.
4. **Demaškaráda:** V prípade, že paket, ktorý je overený kernelom je paket, ktorý príchadza spätne z inej siete, musí byť „demaskovaný“. Pretože ipchains nedokáže plne pracovať s NAT na tento paket sa musí použiť MASQ(Masquerade), tento proces pozmení zdrojovú ip adresu tak aby stanica, ktorá chce komunikovať mimo lokálnu sieť toho bola schopná. Spätný proces je Demaskovanie, ktorý sa vykoná automaticky a zmení zamaskovanú adresu späť na pôvodnu

aby mohol byť paket doručený danej stanici. Paket je po demaskovaní hneď preposlaný na Output chain.

5. **Smerovanie:** Ďalším krokom, ktorým pakety prejdú je smerovanie. Pri tomto procese kernel zisťuje či je paket pre lokálnu sieť(resp. pre tento počítač) alebo cestuje mimo nej(bude preposlaný na Forward chain).
6. **Lokálne procesy:** Paket určený pre lokálny počítač, môže byť spracovaný procesom, ktorý dokáže pakety prijať a odosielať(napr. komunikacia medzi FTP serverom a klientom). Po spracovaní je paket odoslaný na Output chain.
7. **lo rozhranie:** Pokiaľ paket pochádza od lokálneho procesu a je určený pre iný lokálny proces, bude priamo preposlaný na Output chain, kde sa jeho interface nastaví na hodnotu lo(loopback rozhranie), potom sa vráti späť cez Input chain kde jeho inteface bude taktiež nastavený na hodnotu lo.
8. **Forward chain:** Paket určený pre stanicu v inej sieti, alebo paket, ktorý nieje pre lokálny počítač(pc na ktorom je aplikovaný ipchain) sa dostane na túto reťaz. Pokiaľ paket smeruje von z lokálnej siete, kernel na neho aplikuje MASQ tým je schopná komunikovať stanica mimo svoju lokálnu sieť. Ak je paket smerovaný k inej stanici v lokálnej sieti, zkontroluje sa, či na neho nebol použitý MASQ, ak ano paket sa demaskuje. V oboch prípadoch je paket odoslaný na Output chain.
9. **Output chain:** Je posledná reťaz v architektúre, musia ňou prejsť všetky pakety smerujúce von zo zariadenia [4].

1.1.2 Príkazy pre správu ipchains

Tri základne reťaze(input,output a forward), ktoré obsahuje ipchains nieje možné vymazať. Je však možné pridavať a odoberať pravidlá v týchto reťazoch. Syntax pre prácu s ipchains je následovný [1]:

```
ipchains <príkaz> <názov_reťaze> <filtrovacie parametre> -j <TARGET>
```

Filtrovacie parametre sú údaje, ktoré kernel porovnáva s hlavičkou paketu. Môžu byť definované na linkovej, sieťovej a aplikačnej vrstve TCP/IP modelu. V tabuľkách 1.1 a 1.2 sa vyskytujú najčastejšie používané príkazy.

Tab. 1.1: Príkazy pre správu reťazi [2][4]

Príkaz	popis
-N	Vytvorenie novej reťaze
-X	Vymazanie prazdnej reťaze
-P	Zmena politiky reťaze
-L	Zoznam pravidiel reťazi
-F	Vymazanie pravidiel z reťaze
-Z	Vynulovanie paketového a bytového čitača

Tab. 1.2: Príkazy pre správu pravidiel [2][4]

Príkaz	popis
-A	Pripojí nové pravidlo do reťaze
-I	Vloži nové pravidlo do reťaze na určité miesto
-R	Zmeni pravidlo na určitom mieste v reťazi
-D	Vymaže pravidlo na určitom mieste v reťazi

Vytvorenie novej reťaze:

```
ipchains -N newchain
```

Následovne môžeme do tejto novo vytvorenej reťaze zadávať pravidlá:

```
ipchains -A input -s 192.168.1.0/24 -i etho0 -j newchain
ipchains -A newchain -p tcp -d 192.168.1.4 smtp -j ACCEPT
```

Príklad poukazuje na vloženie pravidiel do dvoch rozdielnych reťazoch a zároveň znázorňuje aj možnosť vetvenia z jednej reťaze do druhej na základe pravidla. Prvý príkaz vložil nové pravidlo do reťaze input, pravidlo hovorí kernelu nech skontroluje tieto filtrovacie parametre - hľadaj pakety so zdrojovou adresov 192.168.1.0/24 na rozhraní etho0, pokiaľ kernel našiel takýto paket odkazuje ho -j(jump-to) na reťaz newchain. V tejto reťazi je pravidlo, ktoré hovorí kernelu aby paket prijal, pokiaľ odpovedajú tieto filtrovacie parametre - protokol tcp so zdrojovou adresou 192.168.1.4 a protokolom vyššiej vrstvy je smtp. Pretože kernel vykonáva pravidla sekvenčne v každej reťazi, porovnava hlavičku do doby, kým nenajde pravidlo, ktoré by odpovedlo údajom obsiahnutým v hlavičke paketu. Môže sa stať, že hlavička nebude odpovedať ani jednému pravidlu v novej reťazi. Kernel sa potom vráti späť do reťaze input a dokonči porovnovanie s ostanými pravidlami. V momente keď kernel skonči porovnavanie hlavičky so všetkými pravidlami a nenajde pravidlo odpovedajúce udajom v hlavičke, rozhodne o pakete podľa nastavenej politiky reťaze. Politika

jednotlivých reťazí sa nastavuje pomocou parametrov ACCEPT, DENY, REJECT, MASQ, RETURN.

Príkaz -j(jump-to) sa nepouživa len na vetvenie v reťazoch, používa sa hlavne na definíciu <TARGET> podľa, ktorej sa kernel rozhodne čo vykoná s paketom, ktorý mal zhodnú hlavičku s pravidlom. Kernel sa rozhoduje pomocou týchto definícií:

- **-j ACCEPT**
- **-j DENY**
- **-j REJECT**
- **-j MASQ**
- **-j RETURN**
- **-j REDIRECT**

Posledné dve uvedené v zozname REDIRECT a RETURN sú špecialné definície. REDIRECT sa použije v prípade zmeny portu, na ktorý pôvodne paket smeroval. RETURN je navratná hodnota, má zmysel iba v základných reťazach [1][4].

1.1.3 Manipulácia TOS bytu

Ipchains zvláda mimo filtrovania dátoveho toku aj úpravu TOS[1] bytu. Je schopný meniť štyri bity TOS bytu. Ipchains nemôže ovplivňovať prioritu prichádzajúcich paketov, môže ovplivňovať iba pakety odchádzajúce z linuxovej mašiny.

Tab. 1.3: Nastavenie hodnoty TOS bytu pre ipchains [4]

Popis	Hodnota	Možné použitie
Minimálne Oneskorenie	0x01 0x10	ftp, telnet
Maximálna Priepustnosť	0x01 0x08	ftp-data
Maximálna Spoľahlivosť	0x01 0x04	snmp
Minimálna Cena	0x01 0x02	nntp

Zmena priority TOS bytu sa prevedie zadaním nového pravidla do Output chain.

```
ipchains -A output -p tcp -d 0.0.0.0/0 telnet -t 0x01 0x10
ipchains -A output -p tcp -d 0.0.0.0/0 ftp -t 0x01 0x10
ipchains -A output -p tcp -s 0.0.0.0/0 ftp-data -t 0x01 0x08
```

1.2 Netfilter/Iptables

Netfilter je rozhranie, ktoré komunikuje s kernelom Linuxu, sú v ňom uložené pravidlá podľa, ktorých je kernel schopný spracovávať dátový tok a jednotlivé pakety.

[1]TOS byte je rozobraný v kapitole 1.2.2.

Aby mohol netfilter hovoriť kernelu aké pravidlá má aplikovať na pakety, je potrebné mu ich predom definovať. Pre komunikáciu medzi netfilterom a uživateľom sa používa iptables. Netfilter používa háky(hooks), na ktorých sú rozmiestnené jednotlivé pravidlá. Na úrovni iptables sa nazývajú reťaze(chains). Netfilter a iptables sú navrhnuté pre kernel 2.4+. Zatiaľ čo netfilter je už implementovaný priamo v kernely, je potrebné iptables samostane doinštalovať. Pre správnu funkčnosť musíme nastaviť konfiguráciu, iptables pracujú v režime root. Všetky operácie s iptables sú robené pomocou terminálu.Aby bolo možne používať jednotlivé tabuľky, reťaze a pravidlá musíme do iptables nahrať moduly. Nastavenie modulov pre itables je rôznorodé a závisí od nárokov uživateľa [3].

```
CONFIG_IP_NF_IPTABLES
CONFIG_IP_NF_FILTER
CONFIG_IP_NF_NAT
CONFIG_IP_NF_MANGLE
CONFIG_NETFILTER_XT_MARK
CONFIG_NETFILTER_XT_CONNMARK
CONFIG_NETFILTER_XT_TARGET_DSCP
CONFIG_NETFILTER_XT_TARGET_MARK
CONFIG_NETFILTER_XT_MATCH_MARK
CONFIG_NETFILTER_XT_MATCH_CONNTRACK
CONFIG_NETFILTER_XT_TARGET_SECMARK
CONFIG_NETFILTER_XT_MATCH_MAC
```

Iptables okrem filtrovania, prináša nové funkcie ako sú NAT a mangľovanie paketov. Každá z týchto funkcií má svoju tabuľku, ktorá obsahuje vlastné reťaze. Iptables má tieto tabuľky:

- **Filtrovacia tabuľka:** patria jej reťaze INPUT, FORWARD a OUTPUT.
- **NAT tabuľka:** patria jej reťaze PREROUTING, POSTROUTING a OUTPUT.
- **Mangle tabuľka:** obsahuje všetkých päť reťazí INPUT, FORWARD, OUTPUT, PREROUTING a POSTROUTING.
- **Raw tabuľka:** má len dve reťaze PREROUTING a OUTPUT [3].

1.2.1 Filtrovacia a NAT tabuľka

Filtrovacia tabuľka

Tabuľka slúži na filtrovanie paketov vstupujúcich do linuxovej mašiny, je automaticky aplikovaná pokiaľ sa nezvolí iná tabuľka. Podľa pravidiel je paket prijatý - ACCEPT, zahodený - DROP alebo odmietnutý - REJECT.

NAT tabuľka

Tabuľka NAT slúži na preklad zdrojových alebo cieľových adries(osobných na verejné a verejných na osobné). V tejto tabuľke je taktiež možné použiť filtrovanie paketov. K jej použitiu slúžia príkazy definované v poli <TARGET>:

- DNAT - použijeme ho v prípade, pokiaľ paket nemá pôvod z lokálnej siete smerovača(smerovačom je linux) a je určený stanici v jeho sieti. Cieľová adresa bude pozmená podľa zadefinovaného pravidla v reťazi nat PREROUTING.
- SNAT - sa použije v momente, kedy odchádzajúci paket mieri mimo lokálnu sieť, podľa pravidla v reťazi nat POSTROUTING sa pozmení zdrojová adresa paketu.
- MASQUERADE - má podobné použitie ako SNAT, rozdiel medzi nimi spočíva v tom, že pri použití MASQUERADE sa automaticky začnú vyhľadávať IP, ktoré je možné použiť pre maskovanie. Túto funkciu použijeme v prípade dynamického DHCP pre správny chod siete.
- REDIRECT- používa sa na zmenu cieľového portu na iný, podľa pravidiel v nat PREROUTING a OUTUPUT [2][3].

1.2.2 Mangle tabuľka

Tabuľka slúži na mangľovanie ip paketu, mangľovacia tabuľka dokáže pozmeniť informácie v hlavičke paketu na jednej z jej reťazí. Pomocou mangľovania, sa dá meniť v hlavičke TOS, DSCP a TTL hodnota. Mangľovacia tabuľka obsahuje aj systém značkovania paketov, jednotlivé pakety sa dajú značkovať a potom použiť v súvislosti s iproute2 a L7filtrom. Značkovanie spolu s filtrovacími pravidlami sa používa pri roztriedení dátového toku.

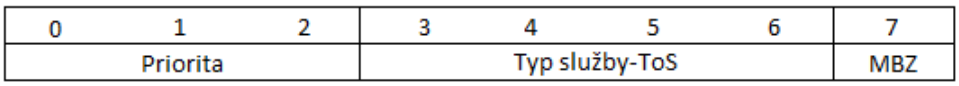
Obr. 1.2: IPv4 datagram [2]

V poli <TARGET> sa používajú príkazy:

- TOS - definuje ho 8 bitov nachádzajúce sa v hlavičke ip paketu. Na obrázku 1.2 môžeme vidieť jeho umiestnenie v hlavičke. Obrázok 1.3 popisuje jeho štruktúru. Priota je trojbitové poľe, ktoré označuje význam paketu. TOS pole má štyri bity a udáva, aký má byť kompromis medzi priepustnosťou, oneskorením, spoľahlivosťou a cenou. Tabuľka 1.4 popisuje možné nastavenia tohoto poľa. Posledné pole je MBZ o veľkosti jeden bit [2].

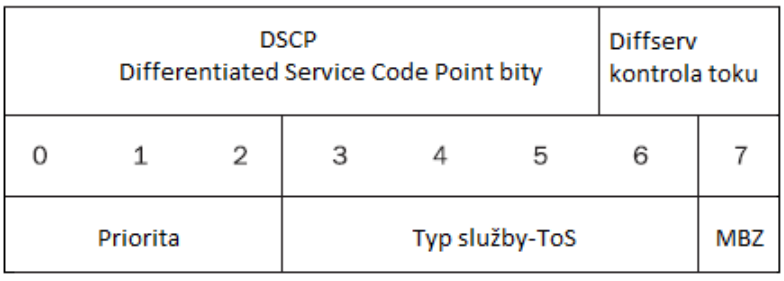
Tab. 1.4: Nastavenie hodnoty ToS bytu v iptables [2]

Popis	Pozícia bitov	Nastavená hodnota
Normálne Služby	0000	0x00
Minimálne Oneskorenie	1000	0x10
Maximálna Priepustnosť	0100	0x08
Maximálna Spoľahlivosť	0010	0x04
Minimálna Cena	0001	0x02



Obr. 1.3: TOS byte [2]

- DSCP - nachádza sa v TOS byte, má hodnotu 6 bitov, čím umožnuje rozmanitejšie nastavenia. Zostavajúce dva bity sú určené na riadenie toku v Diffserv.



Obr. 1.4: DSCP[2]

- TTL - označuje životnosť paketu, maximálna životnosť je 255. Pri prechode paketu routrom za odpočíta -1, keď hodnota dosiahne nulu, bude paket zahodený. Z obrázku 1.2 je vidieť TTL, ktorý zaberá osem bitov z hlavičky paketu.
- MARK - používa sa na značkovanie paketov, takto označkované pakety sú potom rozpoznané nástrojom iproute2. Ten podľa ďalej stanovených pravidiel, ktoré sú mu zadané, spracuje pakety. Značkovanie paketov je veľmi vyznamné pri stavbe HTB a CBQ, kde toto značenie môžeme využiť pre triedenie paketov do front [2].
- SECMARK - používa sa na značkovanie paketov pomocou bezpečnostých značiek, tie sú rozpoznávané v SELinuxe alebo inými bezpečnostnými systémami.
- CONNSECMARK- má podobné použitie ako SECMARK, jedniným rozdielom je, že CONSECMARK sa používa na označenie celého pripojenia [3].

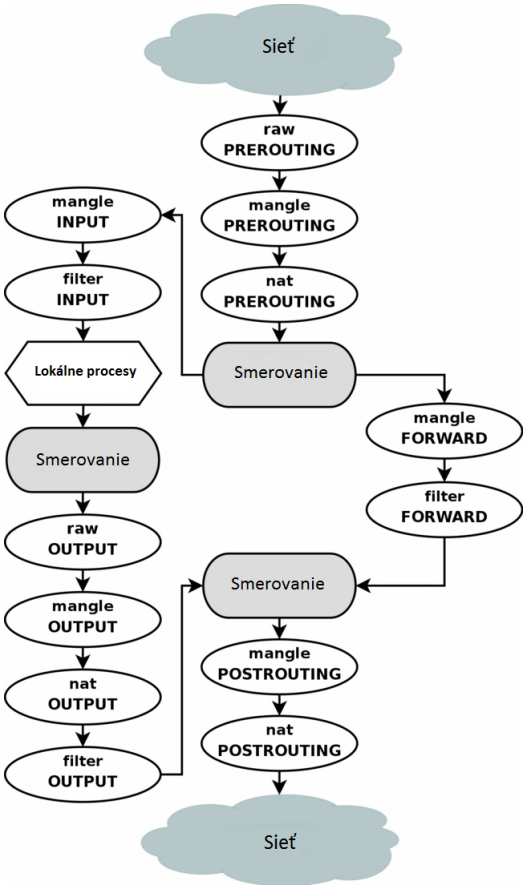
1.2.3 Raw tabuľka

Tabuľk raw sa používa na značenie paketov, ktoré nemajú byť spracované systémom vyhľadavania spojenia - tento systém uchováva informacie o paketoch, vo svojej pamäti, má informácie o zdrojových a cieľových ip adresách, komunikujúcich portoch, použitých protokoloch, stavoch pripojenia a umožnuje používanie NAT. Pomocou týchto informacií je možné zostaviť lepšie filtrovacie štruktúry. Značkovanie sa robí vyplnením poľa <TARGET> s NOTRACK [3].

1.2.4 Architektúra iptables

Obrázok 1.5 popisuje architektúru iptables vytvorenú pre kernel 2.4+. Pakety prichádzajú do linuxovej mašiny, overí sa CRC a stretavajú sa s tabuľkou raw, ta analyzuje reťaz PREROUTING a zkontroluje či prechadzajúci paket neodpovedá pravidlam v jej reťazi. Ak nastane zhoda, na paket je použité pravidlo NOTRACK. Paket následovne prechádza cez tabuľku mangle, v jej reťazi PREROUTING sa hľadajú zhodné pravidlá. Pravidlá v mangle tabuľke a jej reťaziach sú analyzované, bez ohľadu na to, či sa našlo zhodné pravidlo alebo nie, porovnavanie prebieha do doby, kým sa nepreskúmali všetké pravidlá. Následuje tabuľka nat s reťazou PREROUTING, opäť sa porovnavajú pravidlá kým sa nenajde zhoda, pokiaľ zhoda nastane môže dojsť k DNAT, presmerovaniu portov a pod. Paket sa dostáva do situácie kde kernel rozhoduje či je paket pre neho alebo nie. Pokiaľ je paket smerovaný pre neho, dostane sa na INPUT tabuľky mangle, kde môže nastať ďalšia modifickácia hlavičky. Postupne sa paket dostáva k filtrovaciej tabuľke s reťazou IMPUT, zkontolujú sa pravidlá, ak paket nieje zahodený postupuje ďalej na raw OUTPUT(môže opäť dojsť k NOTRACK), mangle OUTPUT(ďalšie možne modifikácie), nat OUTPUT(môže

byť aplikovaný NAT na pakety generované linuxom) a filter OUTPUT(filtrujú sa pakety vychádzajúce s linuxu). Paket sa dostavá na POSTROUTING tabuľky mangle(modifikácie hlavičky) a nat POSTROUTING(môže dôjsť k SNAT) tesne pred tým než opusti architektúru. V situácii, že je paket smerovaný pre inú stanicu, kernel pošle paket priamo na mangle FORWARD a filter OUTPUT [2].



Obr. 1.5: Architektúra iptables [3]

1.2.5 Príkazy pre správu iptables

Iptables nahradil ipchains, čim prevzal aj syntax pre prácu s tabuľkami, reťazami a pravidlami.

```
iptables -t <tabuľka> <príkaz> <názov_reťaze> <parametre> -j <TARGET>
```

Syntaxa iptables sa odlišuje od ipchains pridaním -t <tabuľka>, ktorý nam zadáva do ktorej tabuľky vkladáme novú reťaz alebo pridávame pravidlá. Príkazy pre prácu s reťazami a pravidlami najdeme v tabuľkách 1.1 a 1.2, pre iptables sú totožné až na na príkaz -E, ktorý zmeniť názov vytvorenej reťaze.V tabuľke 1.5 je možno vidieť

používané filtrovacie parametre. Iptables taktiež obsahuje celú škálu <TARGET>, najčastejšie používanými pre tvarovanie dátového toku sú v tabuľke 1.6 [3].

Tab. 1.5: Tabuľka základných parametrov iptables [3]

Parameter	Popis
-p, --protocol	Parameter sa použije keď pri hľadaný určitého protokolu v prenašanom pakete
-s, --src, --source	Parameter na porovnanie zdrojovej adresy
-d, --dst, --destination	Parameter na porovnanie cieľovej adresy
-i, --in-interface	Parameter sa použije pri porovnavaní pravidiel s hlavičkami, na vstupnom rozhraní
-o, --out-interface	Parameter sa použije pri porovnavaní pravidiel s hlavičkami, na výstupnom rozhraní
-f, --fragment	Parameter sa používa pokiaľ hľadame fragmentovaný paket
--sport, --source-port	Parameter sa používa pokiaľ hľadame zdrojový port TCP, UDP a SCTP protokolu
--dport, --destination-port	Parameter sa používa pokiaľ hľadame cieľový port TCP, UDPa SCTP protokolu

Tab. 1.6: Tabuľka <TARGET> iptables [3]

<TARGET>	Funkcia	Popis
ACCEPT		Paket je prijatý
DNAT	--to-destination	Zmení cieľovú adresu
DROP		Zahodí paket
DSCP	--set-dscp,	Nastaví hodnotu DSCP poľa
DSCP	--set-dscp-class	Nastaví DSCP pole na určitú triedu
MARK	--set-mark	Označkuje paket
MASQUERADE	--to-ports	Zmení zdrojovú adresu
NFQUEUE	--queue-num	Pošle pakety do front
REJECT		Odmietne pakety, podľe ICMP správu
SNAT	--to-source	Zmení zdrojovú adresu
TOS	--set-tos	Zmení hodnotu TOS poľa
TTL	--ttl-set	Nastaví hodnotu TTL poľa

1.2.6 L7-filter, IPP2P

L7-filter a IPP2P sú rozšírujúcimi doplnkami nástroja iptables. Obohacujú iptables o možnosť filtrovania dátového toku, na úrovni Aplikačnej vrstvy, kde porovnavajú pravidlá podľa jednotlivých funkčných aplikácií. Fungujú na princípe vyhľadávania pripojenia, preto iptables potrebuje mať nahratý modul ip_conntrack. IPP2P sa od L7 filtru odlišuje väčšiou orientáciou na aplikácie P2P. Nevýhodou týchto filtrou je nadmerné zaťaženie systému, analýzov dát v IP paketu, pri veľkých dátových tokoch. Využitie týchto filtrov, môže byť spojené s označkovaním jednotlivých paketov a roztriedením do front tvarovacích metód. Zápis pravidiel pomocou filtrou môže byť následovný[2]:

```
iptables -I FORWARD -m layer7 --17proto directconnect
iptables -I FORWARD -m ipp2p --dc

iptables -I FORWARD -m layer7 --17proto bittorrent
iptables -I FORWARD -m ipp2p --bit
```

1.3 Iproute2

Iproute2 je balíček programov slúžiaci na pokročilé smerovacie konfigurácie, vytvaranie tunelov a tvarovanie dátového toku. Tvarovanie dátového toku je sofistikovaný systém, použitý ku kontrole šírky pásma. Takúto kontrolu môžeme dosiahnuť pomocou iproute2, ktorý podporuje metódy pre klasifikáciu, prioritizáciu, zdieľanie a obmedzovanie vstupných a výstupných dátových tokov. Nástrojmi sú ip a tc [2].

1.3.1 Nástroj ip

Nástroj slúžiaci na sieťovú konfiguráciu, dajú sa ním nastavovať rozhrania, ARP, politiky smerovania, tunely, dynamické protokoly a pod. Syntax ip je následovný [6]:

```
ip [ OPTIONS ] OBJECT [ COMMAND [ ARGUMENTS ]]
```

1.3.2 Nástroj tc

Pre budovanie QoS architektúr sa používa nástroj tc. Má veľký význam pri stavbe HTB a CBQ, kde slúži pre vytvaranie architektúr, čítaniu označkovaných paketov, obmedzovaniu šírky pásma pomocou jednotlivých pravidiel a filtrovaniu. Podporu jednotlivých metód musíme zaistiť konfiguráciou v kernely. Ako napr. HTB a CBQ:

```
CONFIG_NET_SCHED="Y"
CONFIG_NET_QOS="Y"
CONFIG_NET_SCH_HTB="Y"
CONFIG_NET_SCH_CBQ="Y"
```

Synatax tc je následnovný:

```
tc [ OPTIONS ] OBJECT { COMMAND | help }
```

Selektor U32

U32 je selektor, ktorý sa používa spolu s príkazom tc spolu tvoria filter, podľa ktorého sa dajú filtrovať jednotlivé pakety prechádzajúce linuxom. U32 zadefinuje kernelu šablónu, a ten rozlíši jednotlivé pakety podľa informácií v hlavičkách. Tento selektor v tc funguje podobne ako filtrovanie u iptables[6].

Príklad použitia U32 selektoru:

```
tc filter add dev ath0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
```

2 METODY TVAROVANIA DÁTOVÉHO TOKU

Každý dátový tok prechádzajúci linuxovým prostredím sa môže tvarovať. Tvarovanie pre dátový tok znamená, že oneskoríme doručenie paketov do ich cieľového miesta. Dá sa to dosiahnuť, upravovaním šírky pásma prenašaného dátového toku a pomocou metód, ktoré je kernel schopný používať. Každá metoda dokáže spracovať pakety inak, pakety sú klasifikátorom (podľa nastavených pravidiel, TOS a DSCP, alebo podľa značkovania MARK) triedené do jednotlivých front. Pakety klasifikátor zahodí, prepošle-obíde sa systém spravovania širky pásma, alebo prioritizuje. Pakety vchádzajú do front jednotlivo kde sú následovne usporiadané(preusporiadané) pred výstupom z fronty. Vo frontách sa riadia podľa určitej rýchlosti. Táto rýchlosť ich obmedzuje a zabezpečuje aby mohli použiť len tú širku pásma, ktorá im bola pridelená. Metódy sa dajú z hľadiska štruktúry a použitia rozdeliť na Beztriedne frontovacie disciplíny a Triedne frontovacie disciplíny.

2.1 Beztriedne frontovacie disciplíny pre štandarty 802.3 a 802.11

Beztriedne discilíny označované tiež ako beztriedne qdiscs. Sú základné metody pre tvarovanie dátového toku na celom rozhraní alebo sa nachádzajú vo vnútri listovej triedy, triednej qdics. Môžu pakety oneskoriť a pretriediť pred ich doručením alebo prijať a zahodiť. Každá metoda má svoju vlastnú stanovenú štruktúru, ktorú pakety pri triedení dodržujú.

- **FIFO(pfifo a bfifo):** najjednoduchšia metoda ktorá sa riadí pravidlom prvý prideš prvý odídeš.
- **pfifo_fast:** je predvolená metoda pre všetky rozhrania v Linuxe.
- **Token Bucket Filter(tbf):** spomalí rozhranie na požadovanú rýchlosť. Jeho štruktúra na posielanie paketov, je založená na tokenoch.
- **Stochastic Fair Queuing(SFQ):** spravodlivo rozdeľuje možnosť odosielania dát medzi jednotlivé fronty.
- **Clark-Shenker-Zhang (CSZ):** pretrieďuje pakety a ponúka garantované služby.
- **Priority Traffic Equalizer (TEQL):** prepojí niekoľko internetových spojení do jednoho virtuálneho.
- **Diff-Serv Marker (DS_MARK):** metoda cez ktorú je ISP(provider internetu) schopný ponúkať rozsah sieťových služieb.
- **Random Early Detection(RED) a Generic Random Early Detection(GRED):** používajú sa na chrbticových sieťach(backbone) [6].

2.1.1 First-In First-Out

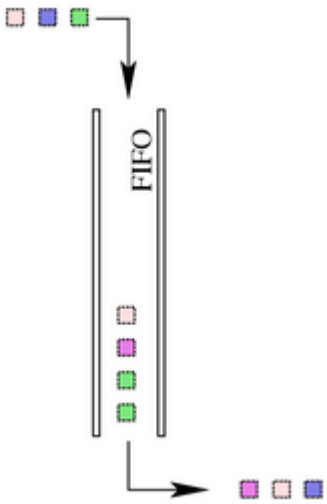
Táto metoda je základnou metódou, od ktorej sa rozvíjali ďalšie. Nedokáže tvarovať alebo roztriediť dátové toky. Je veľmi jednoduchá a obsahuje len jedinú aktívnu frontu, má za úlohu okamžite odoslať všetky pakety z fronty. FIFO sa nachádza vo vnútri každej novo vytvorenej triedy, do doby kým ju nenahradí iná metóda. Každý FIFO qdics má nastaviteľný zasobník, aby nedošlo k zaplaveniu v prípade, keď je počet prichádzajúcich paketov väčší ako počet paketov odosielaných. Zásobník sa nastavuje parametrom *limit*. FIFO má dva druhy použitelnej fronty. Tým je fronta na základe paketov(pfifo) a fronta na základe bytov(bfifo) [5].

Príklad použitia pfifo, zásobník nemôže mať viac ako 45 paketov.

```
tc qdisc add dev ath0 root pfifo limit 45
```

Príklad použitia bfifo, zásobník nemôže obsahovať viac ako 20kB.

```
tc qdisc add dev ath0 root bfifo limit 20480
```

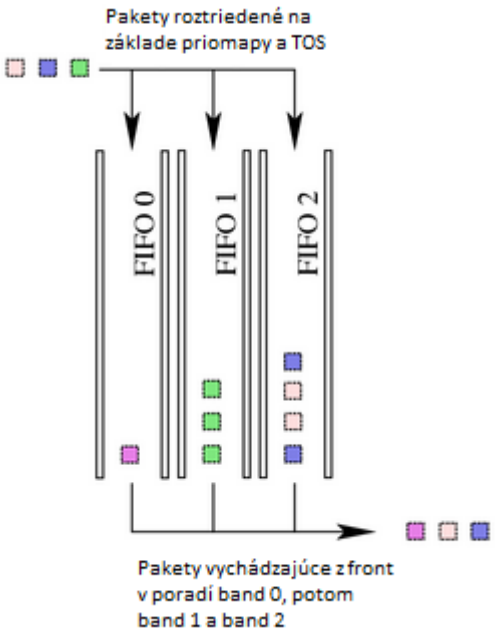


Obr. 2.1: Mechanizmus FIFO [5]

2.1.2 pfifo_fast

Je metoda založenaná na FIFO. Vychádza z pravidla prvý prídeš prvý odideš, narozdiel od FIFO má pfifo_fast tri fronty označované band, kde band 0 má najvyššiu prioritu. Preto ak band 1 a band 2 obsahujú pakety, nemajú povolenie ich vyslať, do doby, kým niesu všetky pakety z band 0 vyčerpané. Metoda pfifo_fast má pevne stanovenú štruktúru, nieje možné ju upravovať alebo pridávať k nej ďalšie fronty.

Pakety na ktoré je aplikovaná táto metoda sú roztriedené do front podľa priomapy na základe TOS bytu. Predovšetkým prioritizuje minimálne oneskorenie do band 0 [5][6].



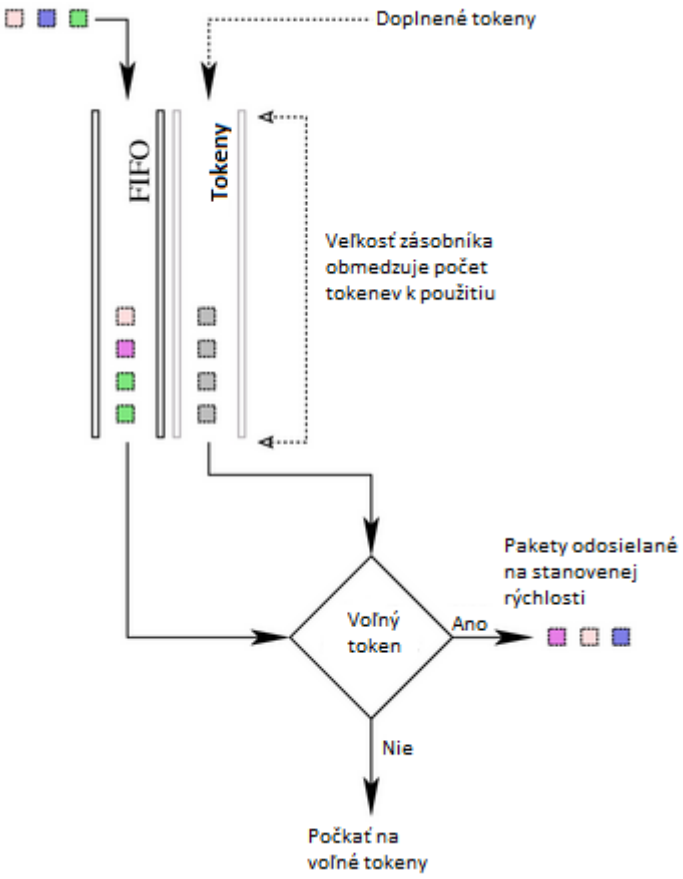
Obr. 2.2: Mechanizmus pfifo_fast [5]

2.1.3 Token Bucket Filter

Metoda založená na predávaní paketov, ktoré prichádzajú s rýchlosťou nepresahujúcu stanovenú rýchlosť. Dovoľuje krátke „strely“ nazývané *burst* pre presiahnutie nastavenej rýchlosti. Architektúra TBF sa zkladá z jednej FIFO fronty a zásobníka(bucket), ktorý je naplnený virtuálnymi znamkami - tokenmi, ktorých veľkosť je udavaná v bytoch. Týmto tokenom sa nastavuje rýchlosť(tokenová rýchlosť). Významným parametrom je veľkosť zásobníka, tá určuje koľko tokenov môžeme používať. Jednotlivé tokeny sú priradené paketom, paket označený tokenom ide von z fronty tokenovov rýchlosťou. Pre príchadzajúce pakety môžu nastať tri možnosti:

1. Data prichádzajú do TBF s rýchlosťou rovnej rýchlosti tokenov. V tomto prípade je paketu okamžite pridelený token a ide von z fronty bez oneskorenia.
2. Data prichádzajú do TFS s rýchlosťou menšiou ako je rýchlosť tokenov. V tomto prípade je vymazaná malá časť tokenov, po vyslaní paketu von z fronty, aby sa nezahltil zásobník. Zvyšné nepoužité tokeny sa použijú na *burst*.
3. Data prichádzajú do TBS s rýchlosťou presahujúcu rýchlosť tokenov. V tomto prípade dôjde k situácií kde sa zásobník dostane nad svôj limit a bude bez

tokenov, TBF sa sám na určitu dobu prišikrtí. Počas tejto doby všetky prichadzajúce pakety budu zahodené.



Obr. 2.3: Mechanizmus TBF [5]

Konfigurovateľné parametre

HTB má niekoľko parametrov, ktoré môžeme podľa našich potrieb jednotlivo nastavovať.

- **limit/latency**
  Limit je počet bytov, ktoré možu čakať vo fronte na voľné tokeny. Latency udáva dobu, ktorú môžu pakety čakať vo fronte. Pri nastavovaní tohto parametru je treba brať v úvahu veľkosť zásobníka, rýchlosť a nastavenie *peakrate*.
- **burst/maxburst/buffer**
  Parametre nastavujúce veľkosť zásobníka v bytoch. Zásobník popisuje maximálne množstvo bytov, ktoré majú tokeny okamžite dostupné.
- **mpu**
  Minimálna paketová jednotka, udáva minimálne bytové použitie tokenu pre paket.

- **rate**
  Nastavuje tokenovú rychlosť vo fronte.
- **peakrate**
  Špecifikuje rýchlosť vyprázdnenia zásobníka, čim určuje aj rýchlosť *burstov*.
- **mtu/minburst** Maximálna odoslaná jednotka, popisuje maximálnu veľkosť paketu, ktorý môže byť odoslaný z rozhrania.

Nastavenie TBF môže vypadať následovne:

```
tc qdisc add dev ath0 root tbf rate 0.5mbit burst 5kb \
                 latency 75ms peakrate 1mbit mtu 1540
```

V tejto kapitole sa použili poznatky z [6].

2.1.4 Stochastic Fair Queuing

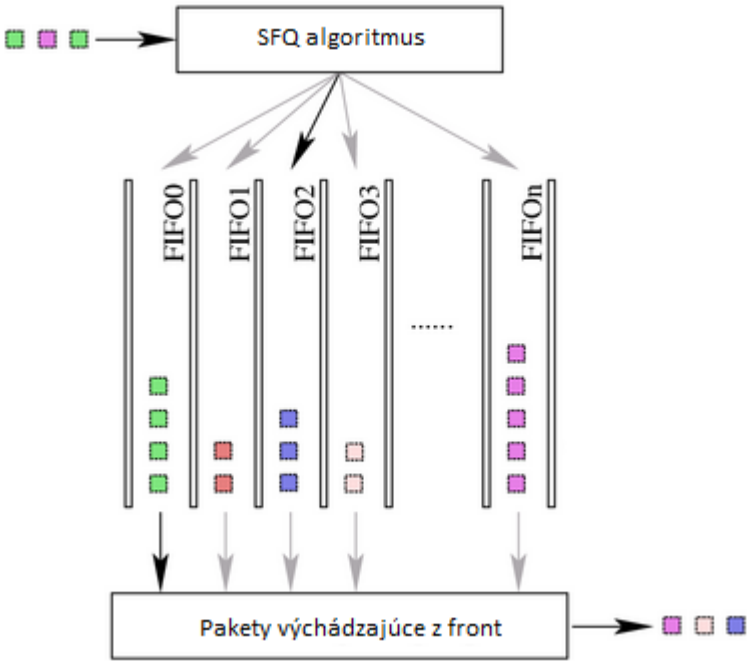
Stochastická metoda, spravodlivo rozdeľuje odosielanie dát z každej fronty. SFQ mán niekoľko FIFO front, do ktorých je dátový tok rozdelený na základe algoritmu používajúceho hash funkciu. Aby nedošlo ku kolízií viacerých datových tokov do jednej fronty kvôli hashu, je hash veľmi často obmeňovaný. Pomocou obmeny sa dosiahne oveľa menšiemu počtu kolízií. Data vychádzajú z front postupne v kruhu(ruletou). Počet bytov, ktoré opustia frontu je možné nastaviť, táto hodnota by mala odpovedať minimálnej hodnote MTU [5].

Konfigurovateľné parametre
SFQ má tri nastaviteľné parametre.

- **perturb**
  Pozmení hash každý pár sekund ktoré mu nastavíme.
- **quantum**
  Množstvo bytov, ktoré je možné vyslať z fronty počas jej doby povoleného vysielanie.
- **limit**
  Maximálne množstvo paketov vo frontách SFQ [6].

Nastavenie SFQ môže vypadať následovne:

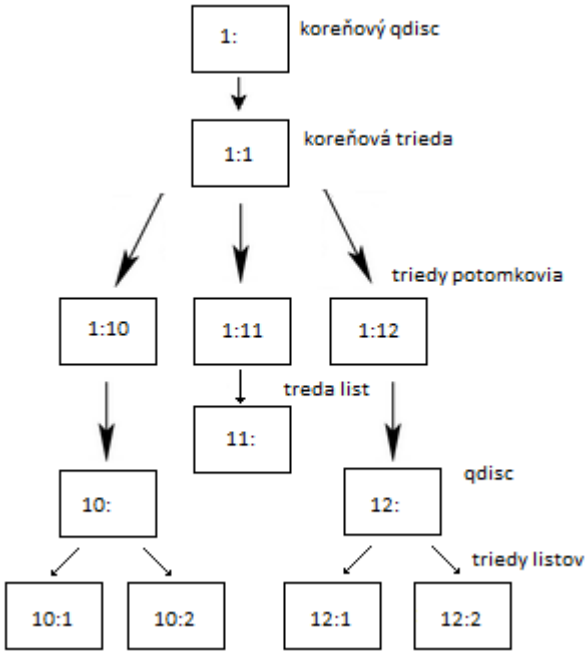
```
tc qdisc add dev ath0 root sfq perturb 10
```



Obr. 2.4: Mechanizmus SFQ [5]

2.2 Triedne frontovacie disciplíny pre štandarty 802.3 a 802.11

Triedne qdisc(classful qdisc) sa používajú v prípadoch kde máme rôzne dátove toky a chceme ich rôzne spracovať. Zkladajú sa z niekoľkých tried, do ktorých sú vchádzajúce dátove toky zatriedené. Aby bolo možné určiť čo spraviť s paketom, musí byť na neho aplikovaný filter. Ten je priamo zadávaný do qdisc, kde na základe uvedeného pravidla rozhodne do akej triedy paket priradí. Podobne každá trieda a následujúca podtrieda, môže mať svoje vlastné pravidla na tvarovanie dátového toku.Triedný qdisc je hierarchický systém usporiadania tried a podtried, povoľuje použitie ľubovolnej beztriednej qdisc. Štruktúra závisí od požiadaviek uživateľa. Rozlišenie jednotlivých tried sa robí systémom <hlavného>:<vedľajšieho> čísla. Pričom sa drží pravidla „1:“ pre koreňový qdics na vrcholu hierarchie. Z obr. 2.5 je vidieť ako sa dajú jednotlive triedy vetviť. Ak bude určitý paket pridelený určitej triede, koreňový qdisc ho pri príchode okamžite pošle na jeho stanovisko. Všetké pakety vstupujúce alebo vychádzajúce z front musia ísť cez koreňový qdisc, pretože jedine sním kernel priamo komunikuje. V momente keď príde požiadavka na vypustenie paketov z fronty niektorej triedy, kernel pošle tuto správu na koreňový qdisc a ten to ďalej preposiela cez svoje triedy. Triedy môžu vyprázdniť svoje fronty, len tak rýchlo ako im povolí ich rodič. Najznámejšími triednymi metódami sú Hierarchical Token Bucket a Class-Based Queuing [6].



Obr. 2.5: Štruktúra triedných qdisc [6]

2.2.1 Hierarchical Token Bucket

Hierarchical Token Bucket(HTB) vychádza z TBF, kde prevzial myšlienku tokenov a zásobníkov. Tento systém neuplatňuje len na jednu triedu ale na celú svoju štruktúru, na triedu sú dva zásobníky jeden je pre tokeny(v rámci triedy listu) a jeden pre ctokeny(v rámci vyšších tried). Tvarovanie jednotlivých dátových tokoch prebieha výhradne len v triedach listov, ostatné triedny slúžia na požičiavanie ctokenov svojim potomkom.Na obr. 2.6, je možné vidieť štruktúru predávania tokenov.

Aby sa ctokeny dali požičiavať, musí byť pod koreňovým qdisc vytvorená jedna koreňová trieda a v nej vnorené zvyšné triedy.Ak by sme mali viac koreňových tried, zdiaľanie pásma by strácalo na efektivite, pretože jednotlivé koreňové triedy si medzi sebou nemôžu požičiavať ctokeny. Potomkovia majú možnosť požičiať si ctokeny od rodičov v momente keď prekročili svoju stanovenú rýchlosť. Potomok vyšle žiadosť na rodičovskú triedy, žiadá ctoken do doby, kým mu nieje priradený alebo kým nieje koreňová trieda dosiahnutá. Tokeny sú požičiavané vzhľadom na hodnotu prírastku *quantum*. Existujú celkovo tri stavy podľa ktorých sa riadi správa tokenov [5][6]:

1. **list < *rýchlosť***
   HTB sa prepne do stavu *HTB_CAN_SEND*, kde rýchlosť zatiaľ nepresiahla stanovenú rýchlosť triedy list. Byty z fronty triedy list budú vyprázdnené s použitím dostupných tokenov do veľkosti *burst* bytov.
2. **list > *rýchlosť* < *ceil***
   HTB sa prepne do stavu *HTB_MAY_BORROW*, kde rýchlosť presiahla sta-

novenú rýchlosť triedy list ale je pod maximálnou rýchlosťou. Trieda list sa pokusi o požičianie ctokenov od svojho rodiča. Pokiaľ sú tokeny dostupné, budú požičané v prírastkoch *quantum* a trieda list vyprázdni svoje fronty do veľkosti *cburst* bytov.

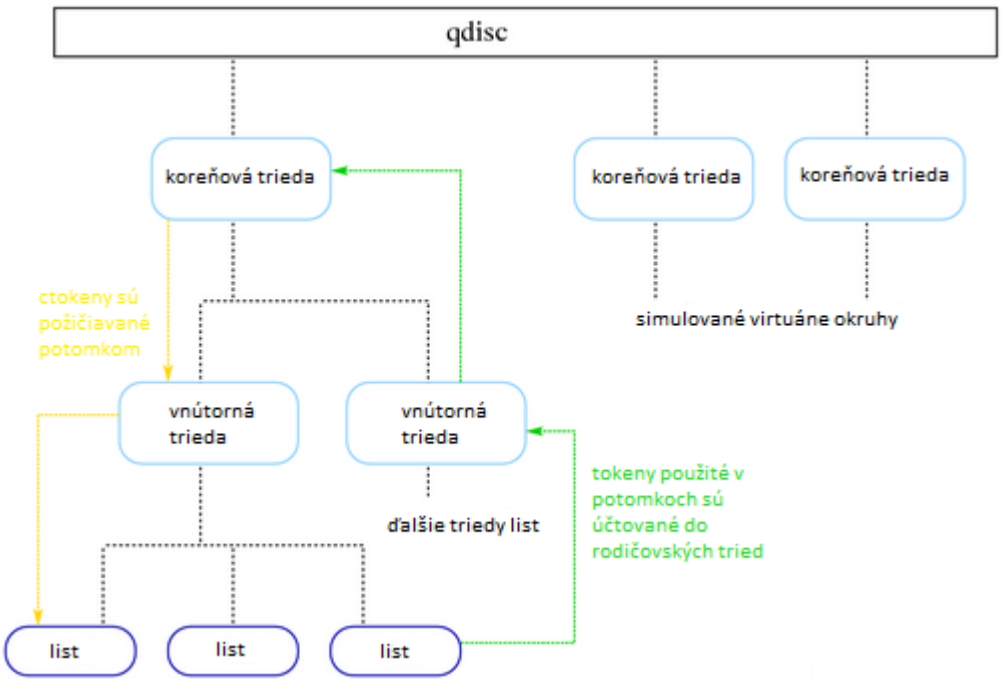
3. **list < *ceil***
   HTB sa prepne do stavu *HTB_CANT_SEND*, kde rýchlosť prekročila rýchlosť ceil. Žiadne pakety nebudú z fronty vyprazdnené. Spôsobí to paketové oneskorenie, zvýši sa latencia čim sa docieli požadovaná rýchlosť.
4. **vnutorná a koreňová trieda < *rýchlosť***
   HTB sa prepne do stavu *HTB_CAN_SEND*, kde rýchlosť zatiaľ nepresiahla stanovenú rýchlosť triedy. Triedy požičiajú ctokeny svojim potomkom.
5. **vnutorná a koreňová trieda > *rýchlosť* < *ceil***
   HTB sa prepne do stavu *HTB_MAY_BORROW*, kde rýchlosť presiahla stanovenú rýchlosť triedy ale je pod maximálnou rýchlosťou. Triedy sa pokusia o požičianie ctokenov od svojho rodiča. Začnú ich posielať svojim súperiacim potomkom v prírastkoch *quantum* na požiadanie.
6. **vnutorná a koreňová trieda < *ceil***
   HTB sa prepne do stavu *HTB_CANT_SEND*, kde rýchlosť prekročila rýchlosť ceil. Vyššie triedy sa nepokúšaju o požičiavanie od svojích rodičov a ani nepožičajú *ctokeny* svojim potomkom.[5]

Konfigurovateľné parametre

- **default**
  Voliteľny parameter spôsobí, aby akýkoľvek nezatriedený tok bol poslaný hardvérovou ryhlosťou, s ignoráciou všetkých tried pripojených na koreňovy qdisc.
- **rate**
  Minimálna želaná rýchlosť slúžiaca na omedzenie prenosu toku.
- **ceil**
  Maximálna želaná rýchlosť slúžiaca na omedzenie prenosu toku.
- **burst**
  Veľkosť zásobníka *rýchlosti*, udáva koľko bytov je maximálne schopné nazbierať za jednu periodu. Pred tým než dorazia nové tokeny HTB vyprázdni *burst*.
- **cburst**
  Veľkosť zásobníka *ceil*. Pred dorazením nových tokenov HTB vyprázdni *cburst*.
- **quantum**
  Kľučový parameter pre kontrolu riadenia systému poožiciavania tokenov.

Tento parameter je si schopný HTB odvodiť sám. Označuje koľko bytov je potrebné obslužiť v triede naraz. Mal by byť čo najmenší ale nie menší ako MTU, pokiaľ by bol menši nebude správne vypočítane množstvo odoberaného pásma.

- **r2q**
  Pomôže vypočítať správnu hodnotu *quantum*. Je tiež automaticky nastavovaný
- **mtu**
  Maximálna paketová veľkosť.
- **prio**
  Určuje prioritu pri prideľovaní voľného pásma. Nižšia číslo má večšiu prednosť [6].



Obr. 2.6: Štruktúra tried a systém požičiavania tokenov [5]

Nastavenie HTB môže vypadať následovne. Vytvorenie štruktúry:

```
tc qdisc add dev ath0 root handle 1: htb default 12
tc class add dev ath0 parent 1: classid 1:1 htb rate 100kbps ceil 100kbps
tc class add dev ath0 parent 1:1 classid 1:2 htb rate 40kbps ceil 100kbps
tc class add dev ath0 parent 1:2 classid 1:10 htb rate 30kbps ceil 100kbps
tc class add dev ath0 parent 1:2 classid 1:11 htb rate 10kbps ceil 100kbps
tc class add dev ath0 parent 1:1 classid 1:12 htb rate 60kbps ceil 100kbps
```

Roztriedenie dátoveho toku pomocou iproute2:

```
tc filter add dev ath0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
tc filter add dev ath0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 flowid 1:11
```

Priradenie qdisc do jednotlivých listov:

```
tc qdisc add dev ath0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev ath0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev ath0 parent 1:12 handle 40: sfq perturb 10
```

2.2.2 Class-based queueing

Class-based queueing(CBQ) je podobne ako HTB zástupca triednych qdisc, ktorá tvorí štruktúru použitím tried. Pomocou ktorých, zabezpečuje rôzne úrovne služieb. Jednotlivé triedy sa delia na triedy listov, vnútorné triedy a koreňovú triedu. V listoch sú rozradené metody na tvarovanie dátového toku, vnútorné triedy slúžia na požičiavanie nezabraného pásma a koreňová trieda reprezentuje celkovú veľkosť použiteľného pásma. Každá trieda má svoju vlastnú, presne stanovenú šírku pásma. Sú si schopné požičiavať nevyužitú šírku pásma pokiaľ sa potrebujú dostať nad svoje nastavené minima, môže taktiež presne určiť, ktoré triedy si možu, a ktoré nemôžu požičiať voľné pásmo.

Spôsob tvarovania CBQ

CBQ pracuje na princípe kde si zistí, či je daná linka dostatočne dlho voľná(voľný čas), aby mohol skutočnú širku pasma znížiť na nastavenú rýchlosť v triede. Voľný čas určí vypočitaním času, ktorý by mal pominúť medzi priemernými paketmi. Efektívny voľný čas sa meria použitím exponenciálneho váženého pohybového priemeru(EVPP), ktorý považuje posledné pakety za dôležitejšie ako tie ktoré pominuli. Z EVPP sa potom odpočíta hľadaný voľný čas, výsledné čislo sa nazýva priemerný voľný čas(PVČ). Pokiaľ má použitá linka PVČ, nulové pakety prichádzajú raz za presne vypočítaný interval. Preťažená linka má negatívne PVČ ak je negatívne moc dostane sa do stavu preťaženie. CBQ v takomto prípade uzavrie linku na určitú dobu. Môže nastať situácia kde voľná linka nazbiera kvôli dlhšej nečinnosti obrovský PVČ, čo by nasledovne dovolilo „nekonečnú“ šírku pásma po niekoľkých hodinách nečinnosti. K zabráneniu tohoto javu je PVČ limitovaný maximálnym voľným časom.

Správanie tried CBQ

Triedy môžu mať rôzne priority. CBQ ich vidí a podľa nich sa správa k triedam rozdielne, priority s nižším číslom majú prednosť pred prioritami s číslom vyšším. Zakaždým, keď príde požiadavok na odoslanie paketov von do siete, spustí sa kruhová obsluha tried tzv. „ruleta“, ktorá začne triedami s najvyššou prioritou. Tie sú potom zoskupené a oslovené pokiaľ maju data k dispozícií. Ak áno vráti sa. Obslúži prvú frontu a postupuje na ďalšiu.

Konfigurovateľné parametre

1. **Parametre pre ovládanie tvarovania**
   - **avpkt**
     Priemerná veľkosť paketu udávaná v bytoch. Je potrebná pre výpočet maximálneho voľného času, je odvodený z *maxburst*, ktorý je udávaný paketmi.
   - **bandwidth**
     Fyzická šírka pásma zariadenia, potrebná pre výpočet voľného času.
   - **cell**
     Čas, ktorý potrebuje paket na odoslanie cez zariadenie môže narásť v závyslosti na veľkosti paketu-nastavuje nespojistosť.
   - **maxburst**
     Počet paketov používaných pre výpočet maximálneho voľného času, tak aby v prípade keď je PVČ na hodnote maximálneho voľného času, mohol byť počet priemerných paketov vystrelený(burst) ,pred tým ako PVČ padne na nulu.
   - **minburst**
     Definuje minimálny počet paketov, ktoré je schopný CBQ prepustiť v prípade preťaženia. V preťažení prepušťa jeden paket za určitu dobu, je však možné sa odmlčať na dlhšiu dobu a poslať všetky tieto pakety.
   - **minidle**
     Nastavuje hodnotu, na ktorú sa reštartuje PVČ v prípadne že by jeho hodnota klesla moc nízko.
   - **mpu**
     Minimálna paketová veľkosť - je potrebná pretože aj paket s nulovou veľkosťou je vložený do 64 bytov na ethernete, tým berie určitu dobu na prenos. Potrebné tiež pre výpočet voľného času.
   - **rate**
     Želaná rýchlosť, ktorou opúšťa dátový tok svoj qdisc.

2. **Parametre pre ovládanie rulety**
   - **allot**
     Určuje koľko bytov môže qdisc poslať z fronty von počas jeho obsluhy ruletou.
   - **prio**
     Nastavuje priority pre jednotlive triedy.
   - **weight**
     Pomáha rulete, určiť koľko bytov dát ma byť poslaných von z fronty. Význam nadobudá u tried, ktoré majú väčšiu šírku pásma. Táto hodnota je potom vynásobnena hodnotou *allot*, čím sa zistí koľko dát môže byť poslaných z fronty von za kolo.

3. **Parametre pre zdieľanie a požičavanie šírky pásma**
   - **isolated/sharing**
     Trieda ktorá je nastavená ako *isolated* nepožičia svoje pásmo príbuzným. *Sharing*, trieda požičia svoje pásmo príbuzným.
   - **bounded/borrow**
     Trieda nastavená ako *bounded* si nemôže požičiať pásmo. *Borrow*, trieda si môže požičiať pásmo.

Vytvorenie štruktúry CBQ:

```
tc qdisc add dev ath0 root handle 1:0 cbq bandwidth 100Mbit \
  avpkt 1000 cell 8
tc class add dev ath0 parent 1:0 classid 1:1 cbq bandwidth 100Mbit  \
  rate 10Mbit weight 1Mbit prio 8 allot 1514 cell 8 maxburst 20     \
  avpkt 1000 sharing
tc class add dev ath0 parent 1:1 classid 1:2 cbq bandwidth 100Mbit  \
  rate 6Mbit weight 0.6Mbit prio 8 allot 1514 cell 8 maxburst 20    \
  avpkt 1000 bounded
tc class add dev ath0 parent 1:2 classid 1:10 cbq bandwidth 100Mbit \
  rate 5Mbit weight 0.5Mbit prio 5 allot 1514 cell 8 maxburst 20    \
  avpkt 1000 bounded
tc class add dev ath0 parent 1:2 classid 1:11 cbq bandwidth 100Mbit \
  rate 3Mbit weight 0.3Mbit prio 5 allot 1514 cell 8 maxburst 20    \
  avpkt 1000
tc class add dev ath0 parent 1:1 classid 1:12 cbq bandwidth 100Mbit \
  rate 4Mbit weight 0.4Mbit prio 5 allot 1514 cell 8 maxburst 20    \
  avpkt 1000 borrow
```

Roztriedenie dátoveho toku pomocou iproute2:

```
tc filter add dev ath0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 match ip dport 80 0xffff flowid 1:10
tc filter add dev ath0 protocol ip parent 1:0 prio 1 u32 \
match ip src 1.2.3.4 flowid 1:11
tc filter add dev ath0 protocol ip parent 1:0 prio 2 flowid 1:12
```

Priradenie qdisc do jednotlivých listov:

```
tc qdisc add dev ath0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev ath0 parent 1:11 handle 30: pfifo limit 5
tc qdisc add dev ath0 parent 1:12 handle 40: sfq perturb 10
```

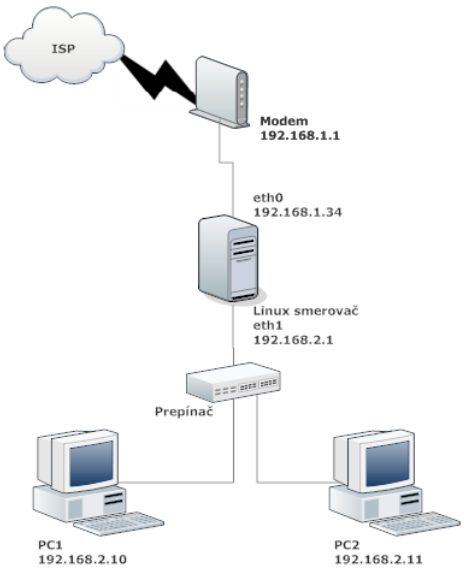
V tejto kapitole sa použili poznatky z [6].

3 PRAKTICKÁ ČASŤ PRÁCE

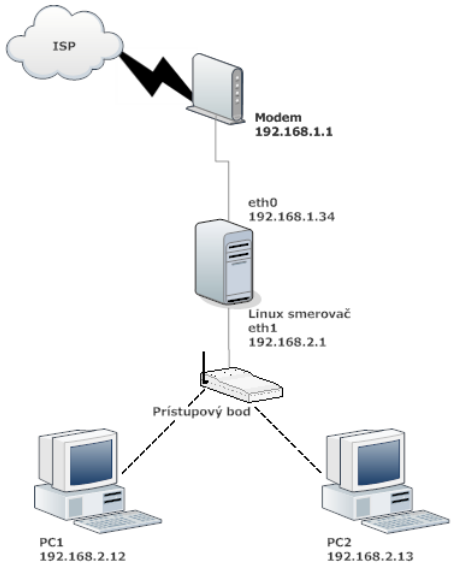
Táto časť sa zameriava na praktickú realizáciu tvarovania dátového toku, pomocou dvoch triednych frontovacích disciplín a následovným aplikovaný na hardvér používajúci štandard 802.3 a 802.11.

3.1 Zostavenie experimentálnej siete

Sieť ktorá bola použitá na tvarovanie dátového toku sa skladala z piatich komponentov - modem, linuxový smerovač, prepínač/pristupový bod a dva koncové počitače. Prepojenie medzi modemom a smerovačom bolo realizované pomocou UTP cat5e kábla. Prepojenie prepínača/pristupového bodu bolo realizované rovnako. Koncové počitače boli prepojené pomocou kablu a rádiového prostredia v závyslosti na použitej technológií.



Obr. 3.1: Zapojenie pre 802.3



Obr. 3.2: Zapojenie pre 802.11

3.2 Linuxový smerovač

Linuxový smerovač disponoval 1,4 GHz CPU a 726 MB RAM pamäťou. Je vybavený dvomi sieťovými kartami, slúžiace ako vonkajšie(eth0) a vnútorné(eth1) rozhranie. Rozhranie eth0 bolo prepojené s modemom, rozhranie eth1 bolo prepojené so sieťou a plnilo úlohu brány.

3.2.1 Konfigurácia Linuxového smerovača

Aby mohol Linux slúžiť ako smerovač a disponovať všetkými prostriedkami, ktoré ma hardvérový smerovač, je ho najprv potrebné správne nakonfigurovať. Konfiguráciu Linuxu môže byť spoužitím distribúcie odlišná. Pre realizáciu bol zvolený Xubuntu, je priamim derivantom Ubuntu. Xubuntu používa miesto klasického rozhrania Gnome, rozhranie Xfce. Xfce je odľahčené grafické prostredie vyznačujúce sa rýchlosťou, stabilitou a nenáročnosťou. Tento OS system plne vyhovoval použitému hardvéru.

Konfigurácia spočivala prvotným nastavením DHCP serveru, a pridelením manuálnych IP adres jednotlivým koncovým počitačom. Pre komunikáciu medzi rozhraním eth1 a eth0 bolo povolené predavanie paketov *packet forwarding* pre IPv4. Pomocou iptables bolo potrebné zadefinovať prijatie všetkej komunikácie smerujúce z rozhrania eth0 na eth1. Komunikácia vnútornej siete s vonkajškom(Internetom) bola zabezpečená použitím NAT[2], pridaním pravidla MASQUARADE. Topológia siete a pridelené IP adresy sú zobrazené na obr.3.1 a 3.2. Kvôli rozsiahlosti je konfigurácia zobrazená v prílohe A.

3.3 Tvarovanie dátového toku - štandard 802.3 a 802.11

Tvarovanie dátového toku bolo realizované metódami vychadzajúce z teoretických poznatkov práce. Boli zvolené metody HTB, CBQ ako rodičovské metody a pfifo vložená v listoch. Tieto metody boli následovne aplikované na linuxový smerovač[3] a použitý štandard. Cieľom meraní bolo určiť odlišnosť metod v závyslosti na použitom štandarde.

Tvarovaná bola šírka pásma o 2 Mbit, ktorá bola následovne rozdelná medzi dve stanice v sieti. Bol vytvorený scénár kde po dobu 30 s bola jedna stanica aktívna, druhá sa po 10 s odmlčala na určitý časový limit a potom opäť začala. Meranie bolo zamerané na jetnotlivé rýchlosti stanic a jitter, ktorý u tvarovacích metod zastupuje dobu paketu stravenu vo fronte.

[2]Možnosti NAT sú rozobraté v kapitole 1.2.1.

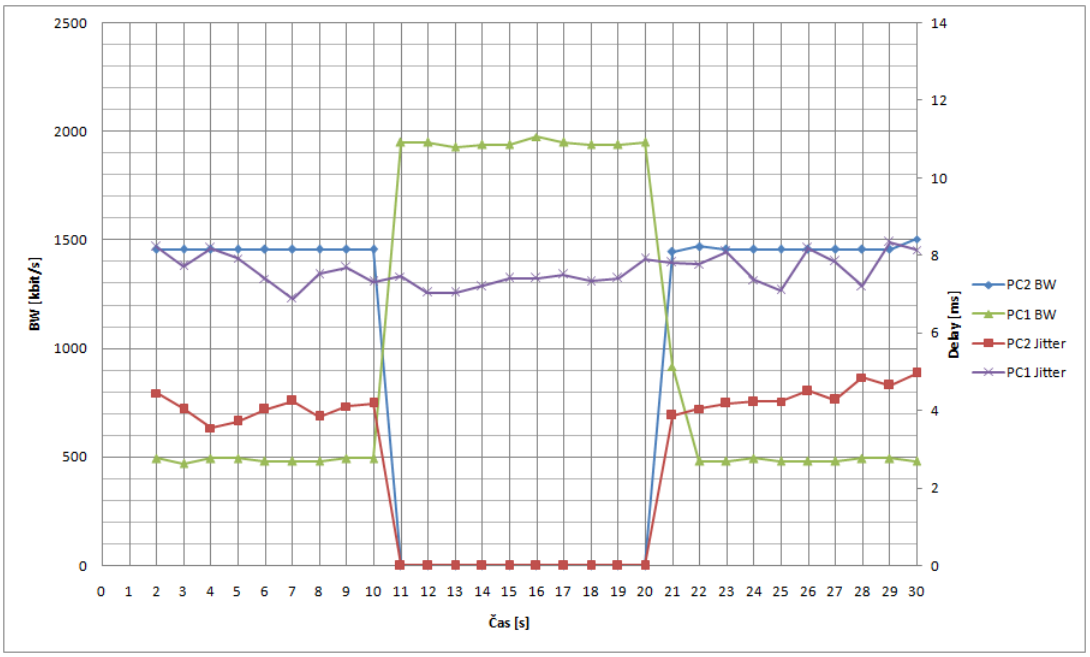
[3]Môžeme tiež uvažovať ako Linuxovú mašinu

3.3.1 Iperf/Jperf

K meraniu bol zvolený iperf/jperf. Jperf je nadstavba iperf, ktorá ma grafické rozhranie vytvorené v jave[4]. Sú založené na bázy klient-server, meranie prebieha generáciou TCP alebo UDP dátových tokov[5] od klienta na server pod určitým portom. Týmto spôsobom je možné zmerať šírku pásma, strátavosť paketov, jitter.

3.3.2 Tvarovanie dátového toku metodou HTB

Pomocou metody HTB bola tvarovaná šírka pásma o veľkosti 2 Mbit/s. Jednotlivé nastavenia boli zvolené tak, aby výstup z metody odpovedal navrhnutému scenáriu. Podrobný prehľad nastavení pre túto metodu ako aj prehľad fitrovania toku sú zobrazené v prílohe B.1 pre štandard 802.3 a prílohe C.1 pre štandart 802.11[6].

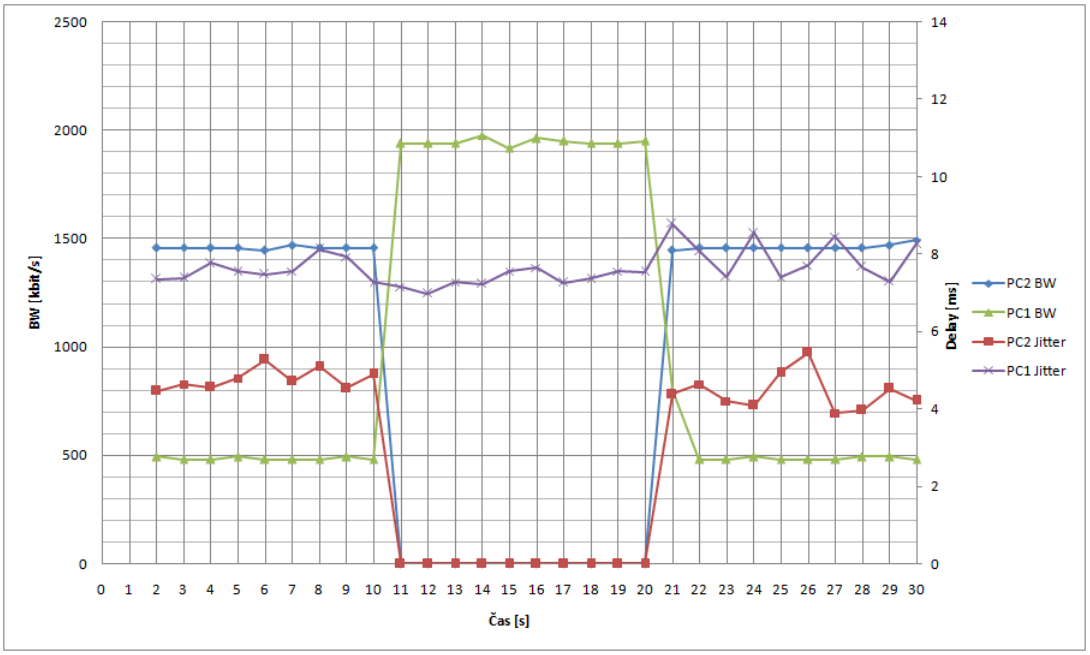


Obr. 3.3: Graf tvarovania dátového toku v HTB štandard 802.3

V grafoch 3.3 a 3.4, je možné vidieť rozdelenie šírky pásma podľa nastavení v prílohach B.1 a C.1. Dátový tok sa rozdelil pre dve stanice, PC1 dostalo 500 kbit/s a PC2 1500 kbit/s. Spoločný súčet oboch stanic dáva do hromady 2 Mbit/s, šírka pásma je maximálne využivaná. Jednotlivé stanice si mohli medzi sebou širku požičiavať čo je možné vidieť v 10 s grafe. Stanica PC2 zastavila prenos, čiže 1500 kbit/s mohlo byť požičané stanici PC1. Šírka pásma sa požičiava v závyslosti na voľných tokenov,

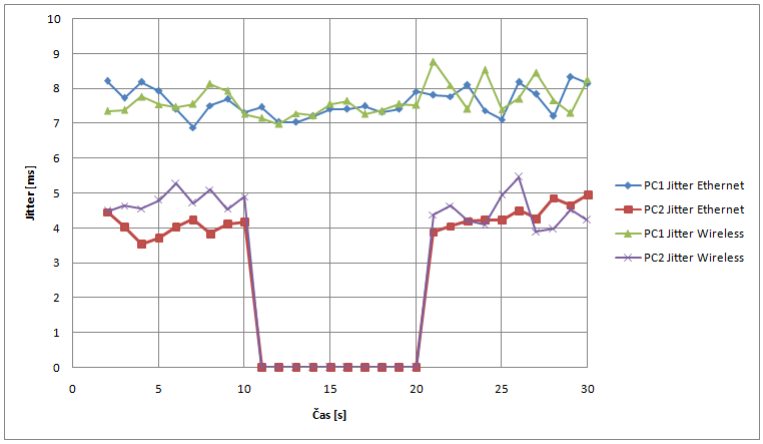
[4]Aby jperf správne fungoval musel xubuntu obsahovať javu.
[5]Iperf/Jperf je schopný generovať aj paralérne dátové toky
[6]Navrh štruktúry HTB a CBQ boli rovnaké pre oba štandardy, rozdiel spočival v IP adresach.

pokiaľ ich ma rodičovská trieda dosť, môže ich požičiať jednej zo staníc, pokiaľ je ta druhá neaktívna. V 20 s času sa stanica PC2 opäť pustila do chodu, rodičovská trieda už nemohla požičiavať svoje tokeny stanici PC1 lebo ich musela davať opäť stanici PC2, ktorá mala garantovanú širku pásma 1500 kbit/s.



Obr. 3.4: Graf tvarovania dátového toku v HTB štandard 802.11

Pridelenie jednotlivých šírok pásma je v oboch štandardoch zrovnateľné. Rozdiel HTB na použitom štandarte je možné pozorovať v jittri. Samostatný jitter je zobrazený v grafe 3.5. Z grafu je možné pozorovať zvýšenie jittru u štandardu 802.11. Celkové zvýšenie činilo 9 % oproti štandardu 802.3, a vzniklo z dôvodu prístupu k zdieľanému médiu jednotlivých staníc. Štandard 802.11 využíva metodu CSMA/CA

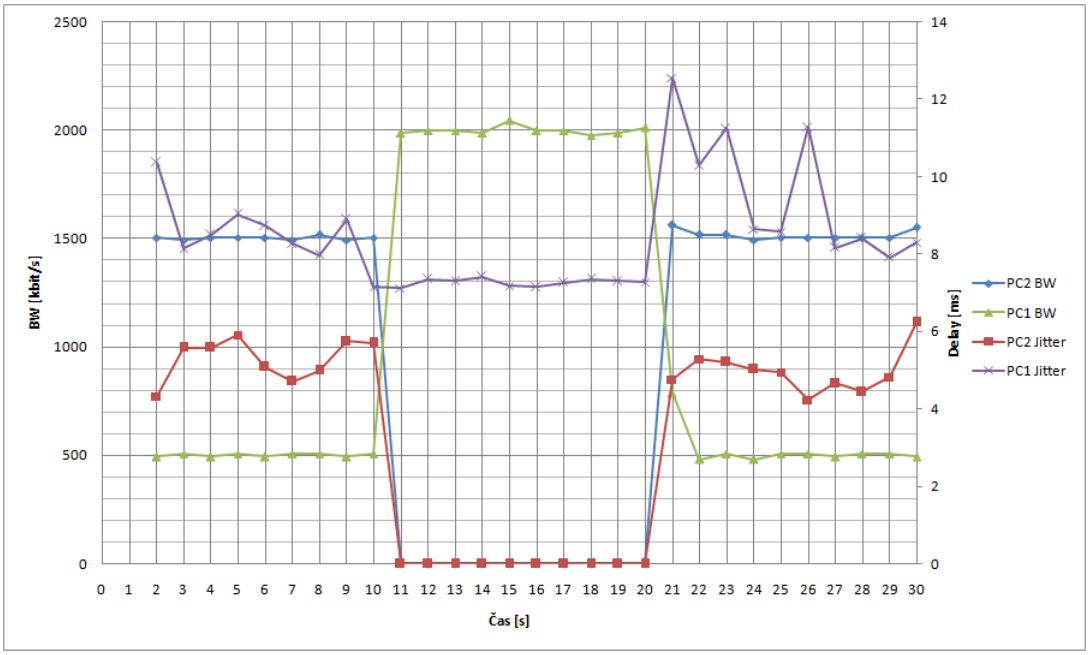


Obr. 3.5: Graf jittru HTB pre štandard 802.3 a 802.11

a DCF, kde stanice, ktoré chcú vysielať musia o zdieľané médium súťažiť vygenerovaním náhodnej časovej periody. Dôsledku tohoto prístupu k zdieľanemu médiu je čas strávený vo fronte väčší u štandardu 802.11 v porovnaní so štandardom 802.3.

3.3.3 Tvarovanie dátového toku metodou CBQ

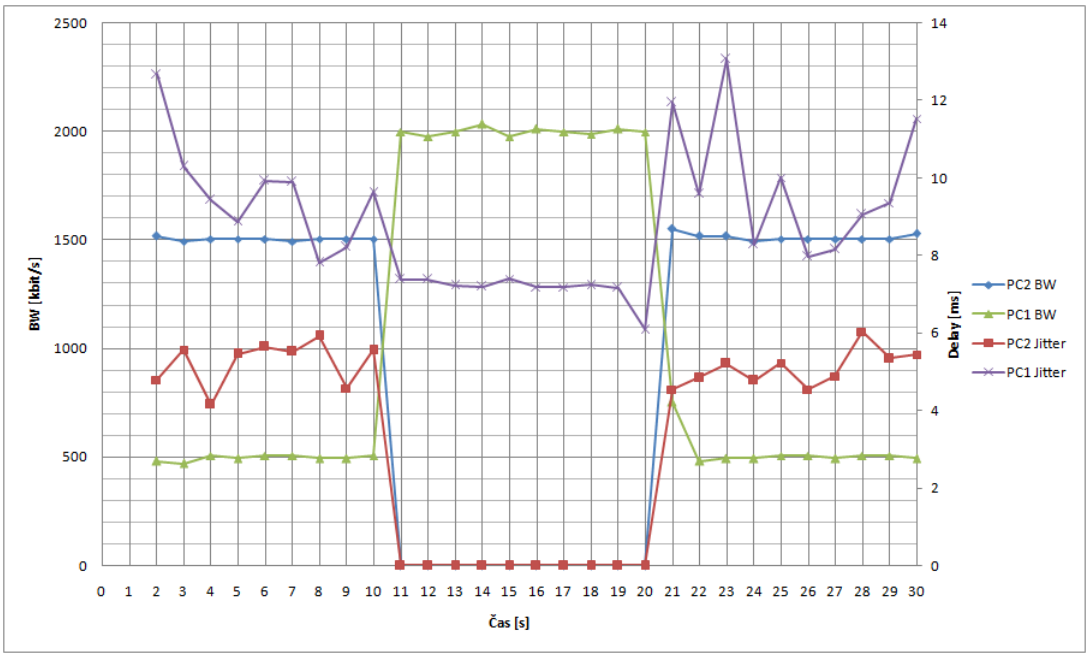
Metodou CBQ bola opäť tvarovaná šírka pásma 2 Mbit/s. Nastavenie metody odpovedalo navrhnutému scenáriu, ktoré je možné vidieť v prílohe B.2 pre štandard 802.3 a prílohe C.2 pre štandart 802.11.



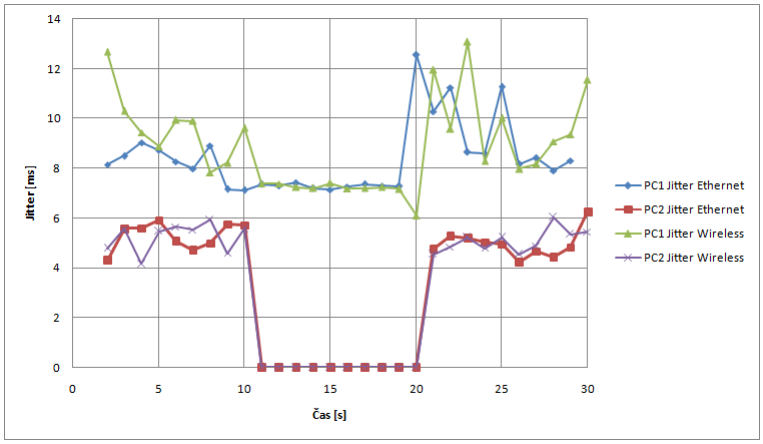
Obr. 3.6: Graf tvarovania dátového toku v CBQ štandard 802.3

Grafy 3.6 a 3.7 zobrazujú rozdelenie šírky pásma podľa nastavení uvedených v prilohách B.2 a C.2. CBQ rozdelil dostupnú šírku pásma, stanici PC1 500 kbit/s a stanici PC2 1500 kbit/s. Šírka pásma bola maximálne využivaná a jednotlive stanice si ju mohli požičiavať, priradenie voľnej šírky je vidieť v grafoch v 10 s kedy stanica PC2 prestala byť aktívna. Jej šírka pásma mohla byť priradená stanici PC1. Šírka pásma u CBQ sa požičiava na dotaze potomka po rodičovi a v závyslosti na globalnej premennej PVČ. Nevyužitú šírku pásma môže na požiadanie svojho potomka rodič požičiavať do doby kým nieje táto šírka pásma vyžadovaná stanicou PC2. V 20 s je stanica PC2 opäť aktívna a rodič prestáva požičiavať jeho šírku pásma stanici PC1. Rýchlosti v jednotlivých štandardoch sa rovnajú. Rozdiel je možné pozorovať v jittri ktorý je samostatne zobrazený v grafe 3.8. Z grafu je možné vidieť zvýšenie jittru u štandardu 802.11. Zvýšenie činilo 6 % oproti štandardu 802.3. Zvýšenie bolo

zapričinené z rovnakého dôvodu ako u metody HTB, a to prístupom k zdieľanému médiu jednotlivých staníc štandardu 802.11.



Obr. 3.7: Graf tvarovania dátového toku v CBQ štandard 802.11



Obr. 3.8: Graf jittru CBQ pre štandard 802.3 a 802.11

4 ZÁVER

Cieľom práce bolo zóznamiť sa s nástrojmi pre správu dátového toku, sjednotlivými metodami pre jeho tvarovanie a aplikovaním tvarovacích metod na hardvér.

Práca bola rozdelená na teoretickú a praktickú časť. V teoretickej časti je možné nájsť tri nástroje, ktoré služia na prácu s datovým tokom Linuxu. Aj keď sa v dnešnej dobe používajú kernely verzií 3.x je v práci spomenutý nástroj ipchains, ktorý sa používal do kernelu 2.4. Dôvodom zaradenia tohoto nástroju do práce je ukážka možností kontroly a tvarovania dátoveho toku na starších verziách kernelov v porovnaní s tými súčasnými. Nástupcom ipchains sa stal iptables, ktorý prevzial časť možnosti úpravy dátového toku od ipchains a rozšíril ho. Z architektúr týchto dvoch nástrojov je možné vidieť obrovský skok v smere tvarovania dátového toku. Kde ipchains disponoval iba filtrovacími možnosťami a jednoduchou úpravou TOS poľa. Iptables rozšíril ponuku možností o NAT a mangľovanie hlavičky paketu. Najvýraznejšou zmenou je možnosť mangľovania, okrem ToS poľa je už schopný meniť aj pole DSCP a TTL. Veľmi dôležitou schopnosťou iptables je funkcia značkovania paketu, kde na základe filtrovacích parametrov určíme pakety k označkovaniu a tie sa dajú po prečítaní týchto značiek metodamy HTB a CBQ zaradiť do príslušnej triedy. Posledným a veľmi dôležitým nástrojom je iproute2. Disponuje filtrovacími možnosťami iptables ale je zásadne používaný na stavbu štruktúr triednych frontovacích metod. Práca ďalej pokračuje preskumaním metod, ktoré je kernel schopný použiť pri tvarovaní dátového toku. Boli rozdelené do Beztriednych a Triednych frontovacích metod, z ktorých boli vybrané metody hodiace sa na tvarovanie toku v ethernete a bezdrátových sieťach. Vybranými metódami sú FIFO, pfifo_fast, TBF, SFQ, HTB a CBQ. Metoda FIFO bola zvolená, pretože je to základná metoda od ktorej sa odvádzajú ostané metody. Jej hlavnou výhodou je jednoduchosť a pritomnosť zásobníka. V podstate netvaruje dátový tok len zaraďuje pakety do svojej fronty a následovne ich posiela von. Metoda pfifo_fast, je ukážkovou metodou, ktorá tvaruje dátovy tok len na základe ToS poľa, je pomerne jednoduchá ale hlavnou nevýhodou tejto metody su práve presne stanovené priority jednotlivých front. Metoda TBF, táto metoda bola preskúmaná, pretože dokáže samostatne tvarovať dátový tok na požadovanú rýchlosť. Jej koncept prideľovania tokenou je jednoduchý a však nevýhodou tejto metody je že nieje možné poslať dáta nad rámec stanoveného zásobníka tokenov. Poslednou popísanou metódou v prvej skupine je SFQ, spravodlivo rozdeľuje odosielania dát medzi jednotlivé fronty, čím má každá fronta možnosť poslať určitý počet bytov von do siete. Výhodou tejto metody je hlavne spravodlivé rozdelenie a roztriedenie dátových tokov do jednotlivých front. Nevýhodou je, že dátový tok je roztriedený pomocou algoritmu čo metodu robí náročnou a samostane netvaruje

dátový tok. Každá z týchto popísaných metód ma jeden hlavný nedostatok. Spracuvávajú dátový tok len na celom rozhraní, preto nieje možne s rôznymi dátovými tokmi zaobchádzať rozdielne. Dá sa to však dosiahnuť pomocou Triednych frontovaciach metod HTB a CBQ. Ktoré majú možnosť vytvorenia vlastne štruktúry a vnoriť do nej jednotlivé metody. Rozdielom medzi HTB a CBQ je predovšetkým v možnostiach ich nastavení, spôsobu odosielania dát a správania jednotlivých tried v štruktúre.

V praktickej časti práce boli použité triedne frontovacie metody HTB a CBQ spolu s metodou pfifo, ktorá služila ako listová metoda. Tieto metody boli aplikované na hardvér a následovne zmerané pod vytvoreným scenáriom pre štandardy 802.3 a 802.11. Výsledky meraní boli vynesené do grafov, kde je možné pozorovať tvarovanie dátového toku na požadované rýchlosti jednotlivých staníc. Spolu s meranými rychlosťami bol aj meraný jitter, ktorý predstavoval dobu stravenú paketov vo fronte metody. Z grafov vyplýva, že jitter bol vyšší u štandardu 802.11, ktorý bol zapríčinený spôsobom prístupu ku zdieľanému médiu. Ani v jednej metode nebolo navýšenie jittru presiahnuté o 10%. Z grafov sa dalo ešte vypozorovať že doby paketov strávených vo frontách metody HTB boli menšie než u metody CBQ. Je to zapríčinené prístupom a spôsobom tvarovania a zdieľania dátového toku u jednotlivých metod. Kde HTB tvaruje a požičiava dátový tok na základe voľných tokenov, CBQ sa riadi na základe, globálnej premennej, merania voľného času medzi prenesenými paketmi, z ktorého následovne odvodí PVČ.

LITERATÚRA

[1] WEHRLE, Klaus, Frank PÄHLKE, Hartmut RITTER, Daniel MÜLLER, a Marc BECHLER. *The Linux® Networking Architecture: Design and Implementation of Network Protocols in the Linux Kernel.* Houston, TX 77098 United States: Prentice Hall, 2004, 626 s. ISBN 0-13-177720-3.

[2] GHEORGHE, L. *Designing and Implementing Linux Firewalls and QoS using netfilter, iproute2, NAT, and L7-filter.* Packt Publishing, 2006. 288 s. ISBN: 1-904811-65-5.

[3] ANDREASSON, Oskar. Iptables Tutorial 1.2.2. *Frozentux* [online]. 2001-2006, 23. augusta 2008 [cit. 2012-11-19]. Dostupné z: <`http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html#GFDL`>.

[4] RUSSELL, Rusty. Linux IP Firewalling Chains. *The netfilter.org project* [online]. 1999-2010, 30. oktobra 2012 [cit. 2012-11-19]. Dostupné z: <`http://people.netfilter.org/rusty/ipchains/`>

[5] BROWN, Martin A. Traffic Control HOWTO. *Guide to IP Layer Network Administration with Linux* [online]. 2006, 29. októbra 2006 [cit. 2012-11-19]. Dostupné z: <`http://linux-ip.net/articles/Traffic-Control-HOWTO/index.html`>

[6] HUBERT, Bert a Spol. Linux Advanced Routing & Traffic Control HOWTO. *Linux Advanced Routing & Traffic Control* [online]. 2002, 19. mája 2012 [cit. 2012-11-19]. Dostupné z: <`http://www.lartc.org/howto/index.html`>

[7] RANKIN, Kyle a Benjamin Mako HILL. *The official Ubuntu server book.* 2. vyd. Upper Saddle River, NJ: Prentice Hall, 2010. 533 s. 2. ISBN 978-013-7081-332.

[8] Iperf [online]. 2013 [cit. 2013-05-03]. Dostupné z: http://iperf.sourceforge.net/

[9] DEVERA, Martin. HTB Linux queuing discipline manual - user guide. *HTB Home* [online]. 2002 [cit. 2013-05-22]. Dostupné z: <`http://luxik.cdi.cz/~devik/qos/htb/manual/userg.htm`>

[10] DEVERA, Martin. Link-sharing and Resource Management Models for Packet Networks. *CBQ (Class-Based Queueing)* [online]. 2008 [cit. 2013-05-22]. Dostupné z: <`http://www.icir.org/floyd/papers/link.pdf`>

ZOZNAM SYMBOLOV, VELIČÍN A SKRATIEK

TOS Type of Service, definuje typ služby v hlavičke IP datagramu

TTL Time to Live, označuje životnosť IP datagramu

DSCP Differentiated Services Code Point, definuje typ DiffServ služby

MBZ Must Be Zero, hodnota nastavená na 0

NAT Network address translation, používa sa na preklad adries

DNAT Destination Network address translation, preklad cieľových adries

SNAT Source network address translation, preklad zdrojových adries

CSMA/CA Carrier Sense Multiple Access with Collision Avoidance, metoda popisujúca prístup k zdieľanému médiu

DCF Distributed coordination function, metoda používajúca sa pre prístup k zdieľanému mediu

ZOZNAM PRÍLOH

A KONFIGURÁCIA LINUXOVÉHO SMEROVAČA

#Inštalácia DHCP serveru
sudo apt-get install isc-dhcp-server

#Povolenie DHCP serveru na rozhranie eth1
sudo nano /etc/default/isc-dhcp-server
”eth1”

#Konfigurácia DHCP serveru
sudo nano /etc/dhcp/dhcpd.conf
ddns-update-style none;
log-facility local7;
default-lease-time 600;
max-lease-time 7200;
option subnet-mask 255.255.255.0;
option broadcast-address 192.168.2.255;
option routers 192.168.2.1;
option domain-name-servers 192.168.2.1;

subnet 192.168.2.0 netmask 255.255.255.0 {
range 192.168.2.10 192.168.2.15;
}

host PC1wired {
hardware ethernet 1c:75:08:eb:36:fb;
fixed-address 192.168.2.10;
}

host PC2wired {
hardware ethernet 48:5b:39:4f:90:71;
fixed-address 192.168.2.11;
}

host PC1wireless {
hardware ethernet ec:55:f9:38:6f:03;
fixed-address 192.168.2.12;
}

```
    host PC2wireless {
hardware ethernet 1c:4b:d6:a9:63:83;
fixed-address 192.168.2.13;
}

#Povolenie predavania paketov
sudo nano /etc/sysctl.conf
net.ipv4.ip_forward = 1

#Konfigurácia rozhrania eth1
sudo nano /etc/network/interfaces
auto eth1
iface eth1 inet static
address 192.168.2.1
network 192.168.2.0
netmask 255.255.255.0
broadcast 192.168.2.255
gateway 192.168.2.1

#Pridanie pravidiel pre komunikáciu s vonkajškom
sudo iptables -t nat -A POSTROUTING -e eth0 -j MASQUERADE
sudo iptables -A FORWARD -i eth1 -j ACCEPT

#Reštart sieťových služieb a DHCP serveru
service networking restart
sudo service isc-dhcp-server restart
```

B NASTAVENIE TVAROVACÍCH METOD PRE ŠTANDART 802.3

B.1 Hierarchical Token Bucket

```
#!/bin/bash

#nastavenie root qdisc
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:10 htb rate 2Mbit ceil 2Mbit

#Nastavenie pre PC1
sudo tc class add dev eth1 parent 1:10 classid 1:200 htb rate 500kbit ceil 2Mbit
sudo tc qdisc add dev eth1 parent 1:200 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.10
flowid 1:200

#Nastavenie pre PC2
sudo tc class add dev eth1 parent 1:10 classid 1:100 htb rate 1500kbit ceil 2Mbit
sudo tc qdisc add dev eth1 parent 1:100 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.11
flowid 1:100
```

B.2 Class-Based Queueing

```
#!/bin/bash

#nastavenie root qdisc
sudo tc qdisc add dev eth1 root handle 1:0 cbq bandwidth 100Mbit avpkt 2000 cell
8
sudo tc class add dev eth1 parent 1:0 classid 1:10 cbq bandwidth 100Mbit rate
2000kbit weight 200kbit allot 1514 cell 8 maxburst 1 avpkt 2000 bounded

#Nastavenie pre netbook
sudo tc class add dev eth1 parent 1:10 classid 1:200 cbq bandwidth 100Mbit rate
500kbit weight 50kbit allot 1514 cell 8 maxburst 1 avpkt 2000
sudo tc qdisc add dev eth1 parent 1:200 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.10
flowid 1:200

#Nastavenie pre notebook
sudo tc class add dev eth1 parent 1:10 classid 1:100 cbq bandwidth 100Mbit rate
1500kbit weight 150kbit allot 1514 cell 8 maxburst 1 avpkt 2000
sudo tc qdisc add dev eth1 parent 1:100 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.11
flowid 1:100
```

C NASTAVENIE TVAROVACÍCH METOD PRE ŠTANDART 802.11

C.1 Hierarchical Token Bucket

```
#!/bin/bash

#nastavenie root qdisc
sudo tc qdisc add dev eth1 root handle 1: htb
sudo tc class add dev eth1 parent 1:0 classid 1:10 htb rate 2Mbit ceil 2Mbit

#Nastavenie pre PC1
sudo tc class add dev eth1 parent 1:10 classid 1:200 htb rate 500kbit ceil 2Mbit
sudo tc qdisc add dev eth1 parent 1:200 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.12
flowid 1:200

#Nastavenie pre PC2
sudo tc class add dev eth1 parent 1:10 classid 1:100 htb rate 1500kbit ceil 2Mbit
sudo tc qdisc add dev eth1 parent 1:100 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.13
flowid 1:100
```

C.2 Class-Based Queueing

```
#!/bin/bash

#nastavenie root qdisc
sudo tc qdisc add dev eth1 root handle 1:0 cbq bandwidth 100Mbit avpkt 2000 cell
8
sudo tc class add dev eth1 parent 1:0 classid 1:10 cbq bandwidth 100Mbit rate
2000kbit weight 200kbit allot 1514 cell 8 maxburst 1 avpkt 2000 bounded

#Nastavenie pre netbook
sudo tc class add dev eth1 parent 1:10 classid 1:200 cbq bandwidth 100Mbit rate
500kbit weight 50kbit allot 1514 cell 8 maxburst 1 avpkt 2000
sudo tc qdisc add dev eth1 parent 1:200 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.12
flowid 1:200

#Nastavenie pre notebook
sudo tc class add dev eth1 parent 1:10 classid 1:100 cbq bandwidth 100Mbit rate
1500kbit weight 150kbit allot 1514 cell 8 maxburst 1 avpkt 2000
sudo tc qdisc add dev eth1 parent 1:100 pfifo limit 5
sudo tc filter add dev eth1 protocol ip parent 1:0 prio 1 u32 match ip dst 192.168.2.13
flowid 1:100
```