Kontrola rychlosti přenosu dat

Aleš Holík

Bakalářská práce
2007

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky

Univerzita Tomáše Bati ve Zlíně
Fakulta aplikované informatiky
Ústav aplikované informatiky
akademický rok: 2006/2007

ZADÁNÍ BAKALÁŘSKÉ PRÁCE

(PROJEKTU, UMĚLECKÉHO DÍLA, UMĚLECKÉHO VÝKONU)

Jméno a příjmení:	**Aleš HOLÍK**
Studijní program:	**B 3902 Inženýrská informatika**
Studijní obor:	**Informační technologie**
Téma práce:	**Problematika kontroly rychlosti přenosu dat**

Zásady pro vypracování:

Vypracujte literární rešerši na zadané téma.
1. QoS – Kvalita služby
2. Technologie IntServ a DiffServ
3. Protokol RSVP (Resource reSerVation Protocol)
4. Omezování šířky pásma – pfifo_fast, TBF, SFQ, PRIO, CBQ, HTB, IMQ, IFB
Proveďte a popište konfiguraci a otestování na GNU/Linux.

Rozsah práce:

Rozsah příloh:

Forma zpracování bakalářské práce: **tištěná/elektronická**

Seznam odborné literatury:

Linux – Dokumentační projekt, Computer Press, 2003
Sobell, M., G.: Linux–praktický průvodce, ComputerPress, 1999.
Nemeth, E., Snyder, G., Hein, T. R.: Linux – kompletní příručka administrátora. ComputerPress, 2004.
Bigelow S.: Mistrovství v počítačových sítích – Správa, konfigurace, diagnostika a řešení problémů. ComputerPress.

Vedoucí bakalářské práce:	**Ing. Martin Sysel, Ph.D.** Ústav aplikované informatiky
Datum zadání bakalářské práce:	**13. února 2007**
Termín odevzdání bakalářské práce:	**24. května 2007**

Ve Zlíně dne 13. února 2007

prof. Ing. Vladimír Vašek, CSc.
děkan

doc. Ing. Ivan Zelinka, Ph.D.
ředitel ústavu

ABSTRAKT

Tato práce přináší přehled možností kontroly síťového provozu v operačním systému GNU/Linux (dále jen Linux). Budou představeny základní přístupy normované jako Integrated a Differentiated service a popis algoritmů, které se používají při kontrole provozu.

Značná část textu bude věnována popisu uživatelského programu `tc`. Programu, který slouží pro kontrolu síťového provozu v Linuxu.

Na konci práce je uvedeno několik konktrétních konfigurací síťového rozhraní.

Klíčová slova:

kontrola provozu, kvalita služby, Linux, řazení do front, klasifikace provozu, omezování provozu, HTB, IFB.

ABSTRACT

This work brings survey of possibility of the net traffic control verification in the operating system GNU/Linux (further only Linux). It deals with basic access standards like Integrated and Differentiated services and description algorithms, that are used for the traffic control.

Great deal of the text presents the description of the application program `tc`. The program, which is used for the traffic control in Linux.

There are several configurations of LAN interfaces at the end of the work.

Keywords:

Traffic control, quality of service, Linux, qdisc disciplines, traffic classification, traffic shaping,HTB,IFB.

Za poskytnutí cenných rad
a zkušeností při vypracování
bakalářské práce bych chtěl touto
cestou poděkovat
Ing. Martinu Syslovi, Ph.D.

OBSAH

ÚVOD

Tato práce pojednává o problematice filtrování komunikace na síťovém rozhraní různých síťových zařízení, zejména pak na routerech – směrovačích.

S nárůstem počtu aplikací přistupujících k lokální firemní síti nebo k Internetu se mohou objevit problémy v datové komunikaci, kdy dochází k výpadkům spojení, zpomalení síťových aplikací a v neposlední řadě může dojít také k nefukčnosti síťové aplikace. Administrátoři těchto sítí pak mají několik možností podobné problémy řešit - a to zvýšením propustnosti sítě, která bývá mnohdy nejen finančně, ale i technicky náročná, nebo mají možnost komunikaci na síti zefektivnit pro provoz kritických aplikací.

Podobně pro poskytovatele připojení k Internetu - ISP (z angl. Internet Service Provider) - je důležité mít nějakou možnost omezovat a kontrolovat své klienty, ať už z důvodů ekonomických (každý ISP má svoji cenovou politiku), nebo z důvodů provozních (filtrování nelegálního obsahu, provozu peer-to-peer sítí, atd...).

Práce nabízí možnosti řešení podobných problémů a to nasazením technik pro kontrolu provozu na "úzkých" místech v dané části sítě.

Práce rozebírá široké možnosti využití operačního systému GNU/Linux, který je díky svým vlastnostem v oblasti směrovačů často nasazován.

I. TEORETICKÁ ČÁST

1 QOS

Je výraz, který se často vyskytuje v literatuře věnující se správě síťových uzlů, routerů a firewallů. QoS je zkratka z anglického **Q**uality **o**f **S**ervice, po překladu tedy – **KVALITA SLUŽBY**.

Počítačová síť založená na protokolu IP (Internet Protocol), tedy i síť Internet, má velmi jednoduchou strukturu. Pakety mají zdrojovou a cílovou adresu a na základě toho mohou projít celou sítí nezávisle na směrovačích a dalších zařízeních. Protokol IP zabezpečuje jen adresování a tím vlastně zaručuje každému paketu nezávislost na ostatních paketech. Tím se může stát, že pakety putují jinými cestami, nebo se přímo zdvojují. Routery mohou zahodit paket bez jakéhokoliv upozornění odesílateli. Aplikace se tedy musí spoléhat na vyšší vrstvy, například na protol TCP.

S rostoucím počtem nabízených služeb náročných na datovou propustnost – streamované video, IP telefonie, ale také P2P sítí je zavedení QoS téměř nutností.

Kvalitu služby lze definovat následovně: Kvalita služby je sada opatření, která se snaží zaručit koncovému uživateli doručení požadovaných dat v požadované kvalitě. Kvalitu služby můžeme také definovat dle [1] a [4] pomocí několika kritérií:

- **dostupnost (minimalizace času výpadku spojení)**
- **chybovost**
- **propustnost**
- **ztrátovost paketů**
- **doba vytvoření spojení**
- **rychlost**
- **detekce a oprava chyb**

Dále v textu budou popsány jen některé aspekty kvality služby, a to takové, které ovlivňují kvalitu služby nejvíce. Jedná se především o šířku pásma, dobu odezvy a jitter.

Šířka pásma znamená rychlost přenosu dat. Šířka pásma nicméně není kritickým faktorem pro řadu aplikací, jako např. pro VoIP telefonii, nebo internetová rádia, tedy interaktivní multimediální aplikace, kde je důležitá především **doba odezvy**, ale dokáže zvýšit kvalitu služby např. při prohlížení webových stránek, odesílání a přijímání elektronické pošty,

či stahování většího objemu dat (např. obrazy instalačních médií OS Linux ve formě ISO souborů a jiné...). Přidělování šířky pásma není obtížné zaručit, důležité je přidělování šířky pásma využít v místě „úzkého hrdla“, tzn. v místě, kde je potřeba šířku pásma omezit.

Jak bylo uvedeno výše, pro interaktivní služby je kritickým faktorem **doba odezvy,** neboli latence. Jedná se o dobu mezi odesláním a přijetím paketu. Jedná se o zpoždění na výstupním zařízení a dále na celé přenosové cestě. Latence u síťového zařízení - routerů je jeho příspěvek do celkové latence paketu jím procházejícího, tedy doba mezi přijetím a odesláním paketu. U přenosu ISO souborů vůbec nevadí, že jsou pakety doručeny i s několikasekundovým zpožděním. V tomto případě hraje významnější roli výše uvedená šířka pásma - je rozdíl jestli je přenos realizován rychlostí 512Kb/s nebo 2Mb/s. Naproti tomu u VoIP telefonie zpoždění jedné sekundy může způsobit výpadek i několika písmen vysloveného slova. Tento druh služby není náchylný na šířku pásma, je možné ji provozovat na relativně pomalých linkách, ale právě na dobu odezvy. Na většině systémů se problém latence řeší přiřazením vyšší priority požadovaným typům dat.

Dalším faktorem ovlivňujícím kvalitu služby je **jitter.** Hodnota jitteru zachycuje velikost proměnlivosti příchozích časů datagramů od vysílače. Vysílací strana posílá datagramy v pravidelných intervalech. Ideálně by přijímací strana měla přijímat datagramy také v pravidelných časových okamžicích, v tomto případě by velikost jitteru byla nulová. Ale mnoho různých zařízení může určité datagramy v datové síti zpomalit, a tak některé datagramy přijdou dříve a jiné mnohem později. Jestliže „pomalé“ datagramy jsou přijmuty příliš pozdě, jsou pak vyřazeny, aby uvolnily místo datagramům, které následují za nimi. Jedna metoda potlačení variability příchodu datagramů vkládá jitter buffer mezi síťovou vrstvu a VoIP aplikaci. Jitter buffer uchová datagramy na přijímací straně. To může vyrovnat proměnlivost příchodu datadramů a tedy umožní použít datagramy, které jsou přijmuty mimo pořadí. Poté pošle příchozí datagramy aplikaci ve správném pořadí. Jitter buffer podrží datagramy po určitý čas ve vyrovnávací paměti, aby provedl potlačení jitteru, což zvýší celkové zpoždění. Další problémy nastanou, když jitter buffer přeteče a další příchozí pakety jsou ztraceny.

Ztráta dat je posledním rozhodujícím faktorem ovlivňujícím kvalitu služby.

Datagramy, které jsou ztraceny a nemohou být obnoveny, vytvářejí v konverzaci mezery. Pokud je ztráta datagramů rozložena náhodně nevede to k tak významnému zhoršení

hlasové kvality. Malé mezery nevadí, ale vysoká ztrátovost datagramů nebo ztráta většího množství dataramů následujících za sebou vede k výraznému zhoršení kvality řeči. Právě ztráta většího množství dataramů následujících za sebou nejvýznamněji zhoršuje kvalitu hovoru. Tento efekt má za následek mnohem větší zhoršení v kombinaci s vysokým zpožděním.

Tabulka hodnot síťových parametrů ovlivňujících QoS:

Parametr sítě	**Dobrá**	**Akceptovatelná**	**Nevyhovující**
Zpoždění	0-150ms	150-300ms	nad 300ms
Jitter	0-20ms	20-50ms	nad 50ms
Ztrátovost	0-0.5%	0.5-1.5%	nad 1.5%

Tabulka 1 - parametry ovlivňující QoS

2 KONTROLA PROVOZU

Kontrola provozu je dle [1] základním obecným nástrojem k ovlivnění a dosažení kvality služby. Jedná se vlastně o celou cestu mezi vstupním a výstupním síťovým zařízením. Základní rozhodnutí jsou:

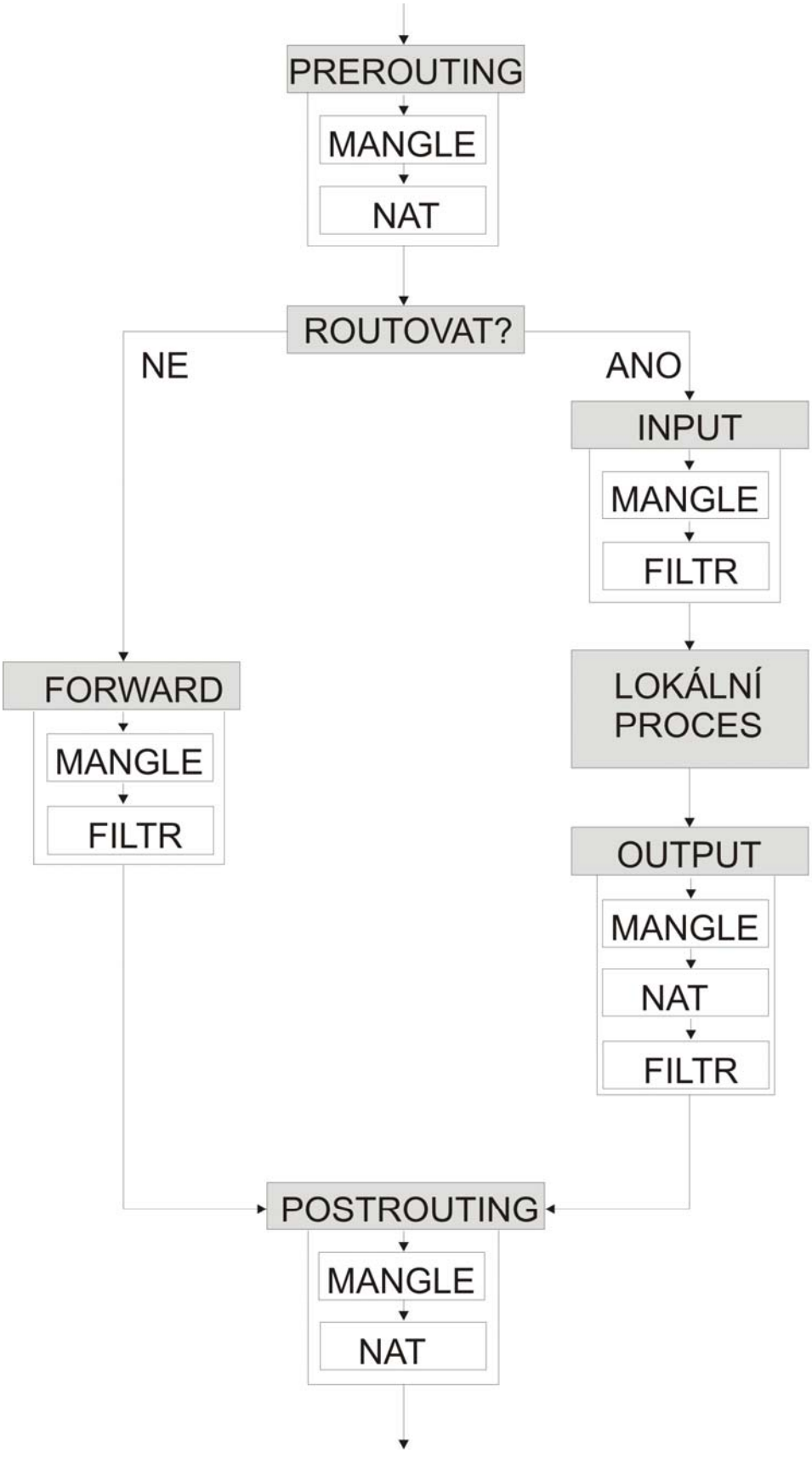
- paket je zahozen na vstupním zařízení
- paket je určen pro tento uzel a je předán vyšším vrstvám
- paket je určen k odeslání, vybere si příslušné výstupní zařízení
- paket je zahozen nebo odeslán dle nějakého plánovacího algoritmu

Ovlivňování paketů na vstupním zařízení se děje málokdy. V podstatě to ani nemá smysl, lepší by jistě bylo neodeslat paket z předcházejícího uzlu, čímž by se ušetřily síťové zdroje.

K druhému bodu lze říci, že uzly se rozhodují na základě cílové adresy. Nic jim ale nebrání aby obsloužily i jiné pakety, či přenechaly paket určený pro sebe dalšímu uzlu. To je výhodné například, když více serverů obsluhuje stejnou službu kvůli rozložení zátěže.

Třetí bod je doménou směrovačů.

Čtvrtý bod představuje plánování odesílaní paketů. Paket je již přiřazen konkrétnímu síťovému rozhraní a čeká na odeslání v nejaké obecné frontě. Plánovací algoritmus určí, v jakém pořadí budou pakety odeslány, případně které z nich budou zahozeny. Existuje několik základních plánovacích algoritmů, od prosté fronty typu FIFO až po složité hierarchicky uspořádané struktury HTB, CBQ. Tuto oblast vymezuje i několik norem a doporučení, nejznámějšími jsou **Differentiated Services** a **Integrated Services**.



Obrázek 1 - Cesta paketu směrovačem

3 INTSERV, DIFFSERV, RSVP

3.1 Technologie IntServ

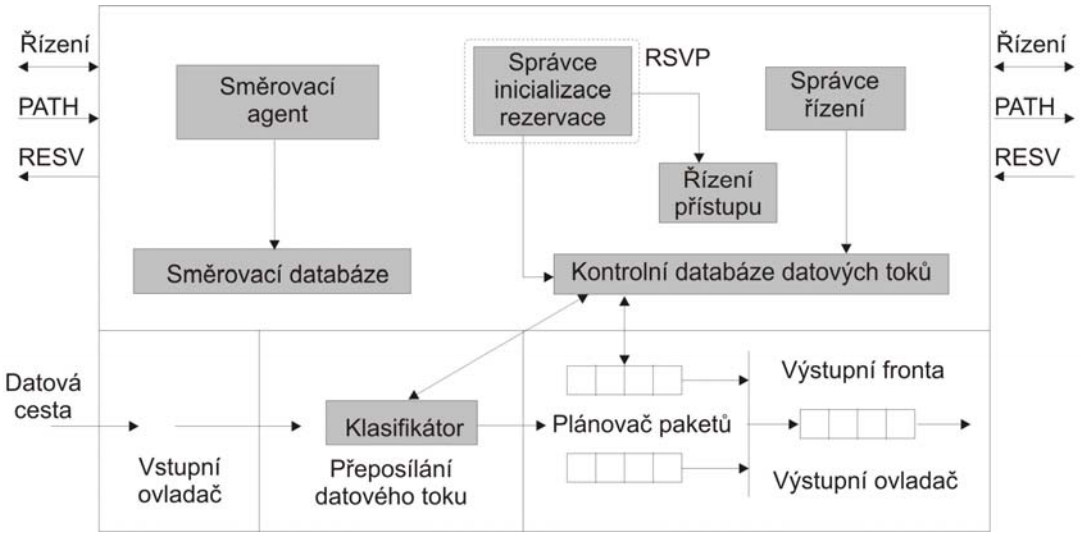
Dle [2] model Integrovaných služeb, dále již Intserv, byl první model, který měl za úkol zajistit požadovanou kvalitu služeb v počítačových sítí protokolem IP. Tento model byl v roce 1994 definován v dokumentu RFC1633. Implementační rámec Intserv obsahuje tyto čtyři části: **Plánovač paketů** (packet scheduler), **kontrolu přístupu** (admission control), **klasifikátor** (classifier) a **rezervační protokol RSVP** (viz. Obrázek 2). Dále je potřeba definovat tok jako rozeznatelný proud souvisejících datagramů (paketů), z něhož vyplývá aktivita jednoho uživatele, a který (tok) požaduje stejnou kvalitu služeb. Tok se může například skládat z jednoho transportního spojení, telefonního IP hovoru či jednoho proudu videa mezi danými dvojicemi uživatelů.

Základní součásti modelu

Pro správnou funkci Intserv musí být ve směrovačích a hostitelích (odesílatel a příjemce QoS) implementovány následující komponenty.

- **Plánovač paketů**
  Plánovač paketů řídí zasílaní různých proudů paketů, používáním souborů front a dalších mechanizmů jako např. časovače. Plánovač paketů musí být implementován v místě, kde jsou pakety řazeny do front. Většinou je každá služba implementovaná ve směrovači, řešena samostatnou frontou a plánovač řídí posílání jednotlivých paketů z front podle předem daných algoritmů.
- **Kontrola přístupu**
  Kontrola přístupu realizuje rozhodovací algoritmus, který směrovač nebo hostitelský počítač používá k určení, zda-li novému toku může být udělena rezervace bez dopadu na dřívější záruky. Kontrola přístupu je spuštěna v každém uzlu, aby mohl rozhodnout, zda bude daná QoS akceptována nebo odmítnuta.
- **Klasifikátor**
  Klasifikátor paketů identifikuje v hostitelích a směrovačích pakety, které budou přijímat určitou úroveň služby. Pro efektivní řízení datové dopravy je každý přichozí paket namapován klasifikátorem do určité třídy. Se všemi pakety, které byly zařazeny do stejné třídy, bude v plánovači paketů zacházeno stejně. Volba třídy je založená na zdrojové a cílové IP adrese a čísle portu či na dalších hodnotách, které musí být přidány ke každému paketu.

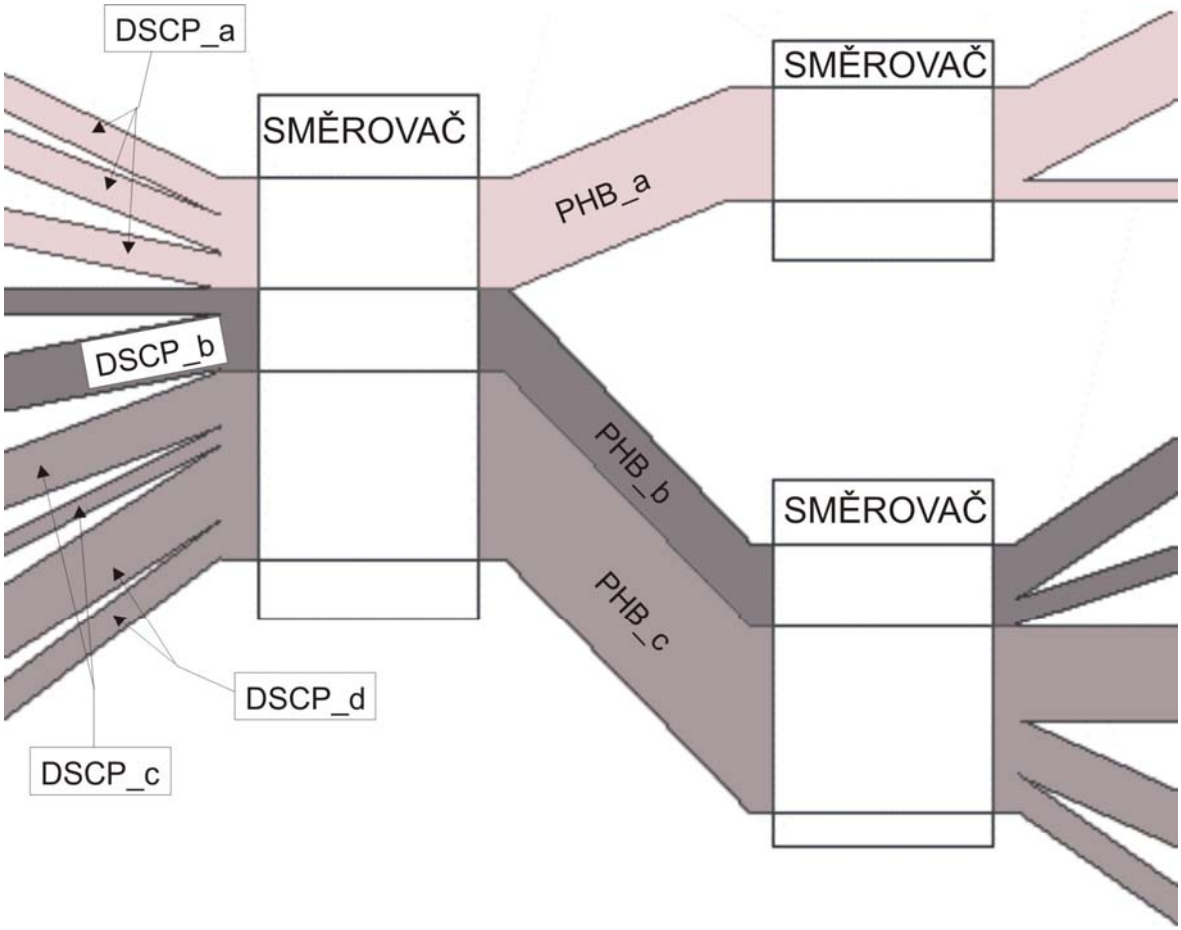
- **Protokol RSVP**
  Čtvrtou a poslední součástí implementace je protokol pro rezervaci zdrojů, který je nutný k vytvoření a udržování stavů v koncových zařízení a ve směrovačích podél cesty toku dat. Tento protokol se nazývá Resource reSerVation Protocol (RSVP).



Obrázek 2 - Technologie IntServ

3.2 Technologie DiffServ

Dle [2] je architektura diferencovaných služeb (dále již jen Diffserv nebo DS) založená na jednoduchém modelu, ve kterém jsou data vstupující do sítě klasifikovaná, upravena na její hranici a poté přidělena k určitému režimu agregace. Každý režim agregace je identifikován DS codepointem. Uvnitř sítě jsou pakety poslány dále podle Per Hop Behavior (PHB) přidruženého k danému DS codepointu.
Základní prvky pro realizaci služeb v DiffServ jsou tzv. Per Hop Behaviour (PHB). PHB představují navenek pozorovatelný režim zasílání použitý v DS uzlech na DS režimy agregace. Obrázek 3, uvedený níže, ilustruje, jak jsou jednotlivé mikrotoky slučovány do PHB a jak PHB mohou být ovládany ve směrovačích. Z obrázku je vidět, že proudy s DS codepointem DSCP_a jsou namapovány na PHB_a, proudy s DS codepointem DSCP_b na PHB_b a proudy s DS codepointem DSCP_c a DSCP_d na PHB_c. To zamená, že více codepointů může být namapováno na jeden PHB.



Obrázek 3 - Technologie DiffServ

Zákazník má nároky na služby, pro které vyjednal dohodu s poskytovatelem služeb. Vyjednané dohody úrovně služby (SLA) specifikují zasílanou službu. Podmnožinou SLA je dohoda o úpravě provozu (TCA), která podrobně specifikuje, jak jsou zákazníkova data ošetřena v prvku úpravy provozu, aby vyhověla vyjednané SLA. TCA obsahuje klasifikační pravidla, dopravní profily, měření, značení a pravidla pro formování datové dopravy, která jsou aplikovaná na dopravní proudy vybrané klasifikátorem.
Úprava provozu v DS doménách je vykonávána v mezních uzlech, tj. v uzlech, které spojují dvě domény. Úpravou v mezních uzlech je vstupní a výstupní doprava zformovaná tak, aby byla přizpůsobena volným zdrojům v cílové doméně. Prvek upravující provoz obsahuje měřiče, značkovače, zahazovače a tvarovače. Tato zařízení měří vstupní datovou dopravu a podle výsledku mohou přeznačit pakety na nový DSCP, zahodit nebo zpozdit přebytečné pakety, tak aby uvedl agregaci do souladu s dopravním profilem specifikovaným v TCA.
Způsoby zasílání v Differv jsou určené hodnotou codepointu v DS poli v IPv4 nebo IPv6

hlavičkách. DS pole v IPv4 hlavičce je tvořeno TOS oktetem a v IPv6 hlavičce oktetem provozní třídy. Každý DSCP je vždy mapován na nějaký PHB.

3.3 Technologie RSVP

Dle [2] RSVP (Resource reSerVation Protocol) je protokol pro reservaci zdrojů na Internetu. Protokol RSVP je používan k získání určité kvality služeb pro jednotlivé aplikační datové proudy nebo toky. RSVP je také užíván směrovači k doručení QoS žádostí všem uzlům podél cesty datových toků a k vytvoření a udržování stavů, které jsou nutné pro poskytnutí požadované služby. V každém uzlu se RSVP pokusí vytvořit rezervaci zdrojů pro daný datový proud.
Některé aplikace vyžadují spolehlivé doručení dat, ale nevyžadují žádné přísné požadavky na včasnost doručení. Ale aplikace jako videokonference, IP telefonie, NetRadio vyžaduje téměř přesný opak; doručení dat musí být včasné, ale ne nutně spolehlivé. RSVP má za cíl poskytnout IP sítím schopnost podporovat odlišné výkonové požadavky různým aplikacím. V protokolu RSVP je datový tok sekvence datagramů, které mají stejný zdroj, cíl a kvalitu služeb. V architektuře QoS založené na protokolu RSVP existují dva základní prvky: zdroje a cíle. Na obou běží RSVP procesy, které se podílejí na protokolu RSVP a vyměňují si RSVP zprávy jménem svých hostitelů. Vyměňují si v podstatě dva druhy zpráv: PATH a RESV zprávy. Zdroj služby pošle PATH zprávu, která je obsažena v IP nebo UDP datagramech. Když je tato zpráva přijata příjemcem služby, tak pokud chce učinit rezervaci pro daný RSVP tok, odpovídá RESV zprávou. Ta je poslána zpět k odesílateli po stejné cestě jako PATH zpráva. V opačném případě je vygenerována RESV ERROR zpráva, která je také poslaná zpět k příjemci. Koncová rezervace je úspěšně ustanovená, když RESV zpráva dosáhne odesílatele a je zpracována ve všech uzlech na její cestě. RSVP je protokol k vyjednání kvality služeb pro konkrétní použití a nejedná se o směrovací protokol. Proto využívá směrovací tabulky ve směrovačích k určení cest k příslušným cílům.

Řízení provozu

Pro zajištění požadované kvality služeb je ve směrovačích implementován mechanizmus nazvaný řízení provozu. Tento mechanizmus má dvě hlavní součásti:

- klasifikátor paketů: určí QoS pro každý paket
- plánovač paketů: docílí slíbenou QoS

Když RSVP žádost přijde do hostitelského počítače nebo směrovače, prochází mechanizmem, který určí, zda-li má uzel dostatek zdrojů k zásobování požadované QoS. Poté je podstoupena policy kontrole, která určí, zda-li má uživatel povolení pro vytvoření rezervace. Jestliže žádost uspěje v obou kontrolách, jsou parametry uložené v žádosti, nastaveny v klasifikátoru paketů. Poté jsou dané parametry nastaveny také v plánovači paketů, který obstará požadovanou QoS. Jestliže v některé kontrole neuspěje, vrací RSVP proces chybu aplikačnímu procesu, který žádost vyslal.

Styly rezervace

Když přijímač pošle RESV zprávu, je v této zprávě také určeno, jak má být s rezervací zacházeno ve vztahu k odesílateli. Tyto soubory voleb jsou nazývány rezervační styly. Tyto styly jsou užívané hlavně pro multicast prostředí. Jestliže je více odesílatelů v jedné RSVP relaci, pak existují dva režimy:

- **Odlišná rezervace (Distinct Reservation)** - vytváří různé rezervace pro každého odesílatele služby
- **Sdílená rezervace (Shared Reservation)** - vytváří sdílenou rezervaci pro specifikované odesílatele služby.

Existuje další volba, která ovládá soubor odesílatelů:

- **Explicitní** - vybere seznam odesílatelů
- **Wildcard** - vybere všechny odesílatele v relaci.

Jestliže kombinujeme tyto módy, pak vznikají tyto styly:

výběr odesílatele	**výběr odesílatele**	
	Odlišná	**Sdílená**
Explicitní	Fixed-Filter (FF) styl	Shared-Explicit (SE) styl
Wildcart	(není definován)	Wildcard-Filter (WF) styl

Wildcard-Filter (WF) styl vytváří jednu rezervaci sdílenou všemi toky od všech odesílatelů. Shared-Explicit styl (SE) vytváří rezervaci sdílenou vybranými odesílateli. Poslední styl je Fixed-Filter (FF), který vytváří rozdílnou rezervaci pro datové pakety od jednotlivých odesílatelů. WF a SE jsou vhodné především pro multicast aplikace.

4 OMEZOVÁNÍ ŠÍŘKY PÁSMA

4.1 FRONTY PAKETŮ, ŘAZENÍ DO FRONT

Dle [3] fronta paketů je pořadí, v jakém budou pakety odeslány ze síťového rozhraní. Způsob řazení do této fronty je právě algoritmus pro zajištění kvality služeb. Protože je prakticky nemožné ovlivnit pořadí dat na vstupu, je problematické uplatňovat QoS pro příchozí traffic. Díky vlastnostem protokolu TCP lze zahazovat pakety, pokud je vyžadováno snížení rychlosti přijímaných dat. To se ale netýká jiných protokolů, kde zahazování paketů nemá na rychlost žádný vliv.

Příchozí traffic na síťovém rozhraní se souhrně označuje **ingress** a jeho tvarování (přeuspořádání, zpomaleni či zahození paketů) se nazává **policing**. V Linuxu lze na ingress uplatnit pouze omezení šířky pásma pomocí zahazování paketů, žádné pokročilé způsoby klasifikace provozu nelze použít.

Odchozí traffic na síťovém rozhraní se označuje **egress** a jeho tvarování se nazývá **shaping**. Pomocí shapingu lze pakety zpomalovat, přeuspořádávat i zahazovat za použití různě složitých algoritmů.

Způsob řazení paketu do fronty je označován **qdisc** (queue discipline). Rozlišují se **classless qdisc** a **classful qdisc.**

4.1.1 CLASSLESS QDISC

Dle [3] classless qdisc nemohou obsahovat jiné qdisc (resp. se to nedoporučuje). Určují pouze, jak bude naloženo s paketem (zda bude zařazen, zdržen nebo zahozen), tedy se jedná o skupinu disciplín, které pouze nějakým způsobem volí co odesílat dříve bez dalšího dělení dat na jednotlivé třídy.

4.1.1.1 PFIFO_FAST

Výchozí fronta [3] pro síťová rozhraní je `pfifo_fast`. Jak název napovídá, jedná se opravdu o frontu FIFO. Pakety jsou odesílány v pořadí, v jakém byly přijaty. Nejedná se tedy o příliš spravedlivou nebo nastavitelnou qdisc, ale částečně se její chování dá ovlivnit. P`fifo_fast` obsahuje tři fronty 0, 1 a 2. Nejprve se vyřizují pakety ve frontě 0, pokud je prázdná, přijde na řadu fronta 1, pokud i ta je prázdná, zpracovává se fronta 2.

Pakety jsou do front řazeny podle příznaku TOS (type of service). Jednotlivé fronty jsou již typu FIFO bez dalších možností nastavení.

TOS (type of service)

Typ služby (Type of Service) je druhý byte v ip paketu [5]. Je určen pro označování paketů, jež mají být doručeny přednostně (s minimální odezvou), či pro označování neprioritního provozu. Tento byte může být využit různě. Jeden způsob využívá bitů 4-7 (počítáno zprava od 1):

Binárně	Decimálně	Význam	pásmo v pfifo_fast /třída v prio 0-2	iptables --set-tos
1000	8	Minimální odezva	0	Minimize-Delay
0100	4	Maximální propustnost	2	Maximize-Throughput
0010	2	Maximální spolehlivost	1	Maximize-Reliability
0001	1	Minimální náklady	2	Minimize-Cost
0000	0	Normální služba	1	Normal-Service

Tabulka 2 - ToS, typ služby

4.1.1.2 TBF (TOKEN BUCKET FILTER)

Algoritmus TBF (Token Bucket Filter) je jednoduchý a nenáročný na výkon [3]. Je vhodný pro jednoduché omezení rychlosti na síťovém rozhraní.

Funguje na bázi "kapes", které plní daty k odeslání a postupně vyprazdňuje danou rychlostí při odesílání dat. Je tak zaručeno, že se nezahltí (pokud je kapsa plná, data se zahazují), ale zároveň nejsou data při krátkodobém větším zatížení zbytečně zahazována. Princip na kterém je založen používá taktéž `htb`. Omezuje provoz na určitou rychlost.

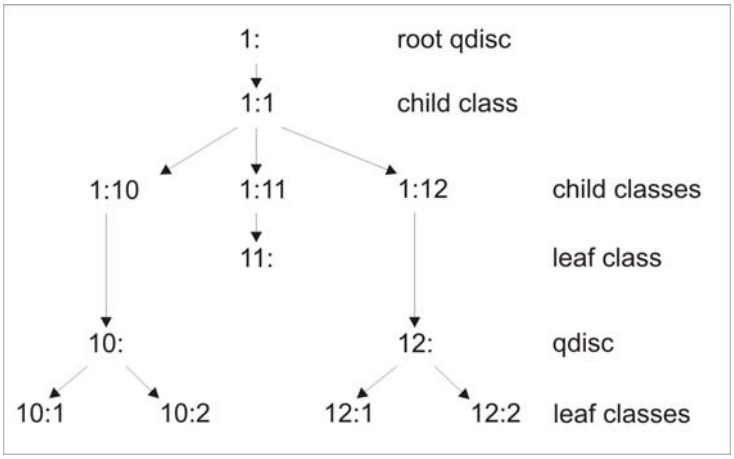
4.1.1.3 SFQ

SFQ (Stochastic Fairness Queueing) je jednoduchý algoritmus pro zajištění rovnoměrného využití sítě všemi aktivními datovými proudy [3]. Nezajistí rovnoměrné využití mezi uživateli či programy, ale např. mezi jednotlivými TCP spojeními vzhledem ke své jednoduchosti docela obstojně. Vytváří více front, do kterých rozděluje jednotlivá TCP spojení a UDP proudy podle hashovací funkce. Tyto fronty jsou zpracovávány algoritmem round robin. Každé spojení má tak šanci na vyřízení a jeden objemný datový proud

nezahltí většinu kapacity linky. Hashovací funkce se v čase mění a jsou tak řešeny kolize (dva proudy ve stejné frontě dlouho nezůstanou).

4.1.2 CLASSFULL QDISC

Když provoz dosáhne classful qdisc [6], je poslán na nějakou z vnořených tříd - musí být "zatříděn". K určení, co se má s roztříděnými pakety dělat, slouží tzv. filtry. Filtry jsou volány prostřednictvím qdisc.



Obrázek 4 - Classfull qdisc

Filtry napojené k příslušnému qdiscu vrací o paketu "rozhodnutí", které je použito k zařazení paketu do jedné z vnořených tříd. Každá podtřída může znova zkusit připojený filtr. Pokud žádný není k dispozici, zařadí paket do qdisc, který je v ní obsažen. Kromě toho, že classful qdisc obsahují další qdisc, většina z nich také provádí shaping, což je dobré k omezování propustnosti a plánování.

4.1.2.1 PRIO

PRIO [3] je podobné `pfifo_fast`, při vytvoření vytvoří 3 třídy, které mají prioriy 0, 1 a 2 a zpracovávají se postupně (od nejnižší, teprve až je prázdná, může se zpracovat další třída). Jednotlivé třídy zprácovávají pakety úplně obyčejnou frontou, ale lze na ně pověsit libovolnou jinou qdisc.

Počet tříd i klasifikaci lze nastavit, ve výchozím nastavení se používá obdobný způsob klasifikace jako u `pfifo_fast`. Přímo shaping tato qdisc neumožňuje, ale lze docílit pokročilých nastavení prioritního zpracování dat.

4.1.2.2 CBQ

CBQ [3] (Class Based Queueing) je dlouho používaná a na nastavení velmi bohatá qdisc. Podporuje libovolné vnořování tříd a shaping. Třídy mohou půjčovat svou konektivitu podtřídám a získávat od svých předků. Podporuje také priority (obdobně jako PRIO).

Při výpočtech vychází z doby nečinnosti linky, průměrovaných velikostí paketů a šířky pásma. Mezi jednotlivými třídami pak vybírá algoritmem wighted round robin (který lze také ovlivnit nastavením).

4.1.2.3 HTB

Disciplína vzniklá na základě CBQ (Class Based Queueing) [3], má jednodušší a více intuitivní nastavení. Pro jednotlivé třídy ja k dispozici nastavení především dvou parametrů - rate a ceil. Ceil je celkové maximum - HTB neumožní odesílat pakety dané třídy rychleji než je nastaveno. Takže například je možné vytvořit třídu určenou pro smtp protokol, ve které bude omezen ceil na 256kbit/s. Poštovní server pak nikdy nebude moci odesílat data rychleji, než touto rychostí.

Parametr rate označuje jakousi garantovanou rychlost. Nejprve jsou odeslány pakety v rámci rate jednotlivých tříd a poté když něco zbyde, rozdělí se zbytek mezi třídy, co chtějí přenášet data větší rychlostí, než je jejich rate.

4.1.3 SHRNUTÍ

Qdiscy jsou užitečné, ale ne všemocné. Každý je vhodný na něco jiného. Jaké jsou tedy jejich výhody/nevýhody? **TBF** přesně omezuje, ale nerozlišuje co omezuje. Když například uživatel stahuje z dc a zároveň prohlíží www stránky, pak jsou přenosy z dc realizovány rychlostí, na jakou byly omezeny pomocí **TBF**, ale pakety HTTP jsou často nadlimitní a proto se stránky načítají dlouho. Pokud u qdiscu **PRIO** není omezena rychlost uvnitř třídy, pak se méně prioritní třídy nemusejí dostat vůbec ke slovu. Pokud je třída u **HTB, CBQ** přeplněná, pak neovlivní ostatní třídy. Pakety v nich se dostanou postupně na řadu a odešlou se bez zpoždění. Pakety v přeplněné třídě na tom již tak dobře nejsou. Jelikož je tato třída přeplněná jsou některé pakety pozdrženy/zahozeny. Je lepší proto třídy naddimenzovávat.

4.2 VIRTUÁLNÍ ZAŘÍZENÍ

4.2.1 IMQ (INTERMEDIATE QUEUEING DEVICE)

IMQ [7] spočívá ve vytvoření speciálního zařízení (realizované přes modul jádra), do kterého se dají přes mangle tabulku iptables "poslat" data. Toto zařízení se z pohledu shapingu chová jako klasické fyzické zařízení (eth0,..) a tak mu lze přiřadit některou z předchozích disciplín.

Přestože není možné nějak ovlivnit to co druhá strana posílá, je možné regulovat (také se v tomto případě často mluví spíše o policingu než shapingu) příchozí průtok - a to díky vlastnostem TCP protokolu. TCP automaticky snižuje rychlost odesílaných dat v případě že nedostane včas potvrzení o přijetí. Tím, že se uměle omezí průtok na cestě paketu, tak vysílač automaticky sníží rychlost a v důsledku tím poklesne i rychlost přijímaných dat na rozhraní, které je jinak nemožné přímo regulovat.

Bohužel toto nefunguje u nepotvrzovaných protokolů (např. UDP, ICMP). Zde neexistuje zpětná vazba na průchodnost paketů. Proto také jsou tyto protokoly často zneužívány k tzv. floodům. Útočník v tomto případě zaplavuje oběť velkým množstvím těchto paketů. Pokud má oběť k dispozici pomalejší linku, může dojít k jejímu úplnému zahlcení. Proti podobným útokům neexistuje možnost se bránit na úrovni shapingu - jediná cesta bývá upozornění poskytovatele internetu (který tento zdroj zakáže u sebe), případně fyzická eliminice zdroje (může se například jednat o zavirovaný PC, takže řešením je odpojení od sítě, či u bezdrátových technologii zákaz na úrovni asociace k AP).

4.2.2 IFB (INTERMEDIATE FUNCTIONAL BLOCK)

IFB [8] je virtuální zařízení, které je nástupcem virtuálního zařízení IMQ, jež nikdy nebylo díky svým problémům se stabilitou integrováno do linuxového jádra. Nevýhodou IFB je nemožnost použití některých klasifikátorů, o kterých bude řeč dále.

4.3 FILTRY

Po vytvoření tříd, je potřeba je naplnit [5]. O to aby skrz ně procházely správné pakety se starají filtry. Filtry jsou vlastně pravidla. Filtrovat lze pouze v rámci jednoho qdiscu. Například v příkladu pro **HTB** lze filtry směrovat data z 1:0 do 1:31 a nikoliv do 31: a 10:1. Při požadavku směrovat data z 1:0 do 10:1 se to musí udělat postupně z 1:0 do 1:10 a z 10:0 do 10:1. Priorita u pravidel určuje pořadí v jakém se vykonávají. Filtry s číselně nižší prioritou se vykonají dříve, a proto je výhodné zde mít nejspecifičtější pravidla. Filtry se nemusejí mazat, stačí smazat příslušný qdisc. Filtrovat lze podle:

- zdrojové/cílové adresy
- zdrojového a cílového portu
- ip protokolu (tcp, udp, icmp, ...)
- značky z firewallu
- podle pole TOS
- podle různých položek hlavičky IP, TCP, UDP datagramu

Druhy filtrů:

- u32: založen na informacích obsažených v paketu (IP adresa, zdrojový/cílový port ...)
- fw: firewall – založen na „značkování“ paketu firewallem
- ipp2p: dokáže rozlišit p2p sítě, spolupracuje úzce s firewallem
- route: založen na rozhodování, kudy má být paket směrován
- rsvp: filtruje pakety dle RSVP – použitelné v sítích, které má administrátor pod úplnou kontrolou.

V této práci budou dále probrány první dva druhy.

4.3.1 FILTR u32

Filtr u32 je založen na informacích obsažených v paketu [5], s výhodou může být používán pro filtrování pomocí zdrojové/cílové adresy, portu, ale také podle TOS pole paketu.

4.3.2 FILTR fw - firewall

Filtr fw [5] lze použít tam, kde je vyžadováno filtrovat pakety např. podle MAC adresy zařízení, nebo ip adresy, portu, skupiny portů. Oproti filtru u32 ale může být nasazen tam, kde se provozuje SNAT (skutečné zdrojové adresy zůstávají skryty). Tato značka není součástí IP paketu, čili ji nemůžou vidět ostatní počítače narozdíl od TOS. V linuxu se pro filtr fw používá firewall iptables, který je součástí jádra systému. Nevýhodou filtru fw je nemožnost jeho použití s virtuálním rozhraním IFB.

```
#Označení paketu pomocí iptables (podle MAC počítače)
#iptables -t mangle -A PREROUTING -p tcp -m mac --mac
#source 12:34:56:78:90:ab -j MARK --set-mark 1
```

4.3.3 FILTR IPP2P

Filtru IPP2P [5] umí detekovat a provoz sítí typu P2P. Rozeznává protokoly:

- eDonkey
- Gnutella
- FastTrack (Kazaa)
- Direct Connect
- Bittorent

Filtr ipp2p je určen pro firewall iptables. Lze vybrat všechny pakety nebo pouze datové přenosy. Pokud chce administrátor vybrat všechny pakety patřící DC, pak použije `--dc`. Pro data `--dc-data`. Pokud chce jedním pravidlem pokrýt všechny protokoly, pak použije `--ipp2p/--ipp2p-data` (kromě Bittorentu - pro něj musí použít `--bit`). Ipp2p funguje pouze na TCP paketech. Pokud nemá být zakazován p2p provoz, pak bude muset být použit CONNMARK. CONNMARK umožňuje označkovat celé TCP sezení.

```
## Označí všechny datové přenosy p2p sítí, které ipp2p detekuje
# iptables -t mangle -A PREROUTING -p tcp -m ipp2p --ipp2p-data -j
# MARK --set- mark 1
```

II. PRAKTICKÁ ČÁST

5 IMPLEMENTACE V LINUXU

Tato část práce bude pojednávat o implementaci QoS v Linuxu. Bude obsahovat podrobný návod pro zprovoznění QoS na routeru s operačním systémem Linux. Popisovaná konfigurace bude testována na distribuci Debian Etch, nicméně funkční by měla být bez větších problémů na většině distribucí. Kontrola provozu bude implementována pomocí disciplíny HTB s použitím imaginárního zařízení IFB pro odchozí traffic. Pro zprovoznění QoS je potřeba mít:

- jádro se zakompilovanou podporou pro QoS a firewall iptables
- balík iproute2, který obsahuje sadu programů pro práci se síťovým nastavením

5.1 Podpora QoS ze strany jádra systému

Pro většinu disciplín pro QoS - HTB, CBQ, PRIO, TBF... je podpora standardně zakompilována výrobcem distribuce. Určitě je tak tomu u "velkých distribucí" typu Debian, Fedora, Mandriva, SuSe. Nicméně je možné, že u menších distribucí tomu tak nebude.

Úplnou výjimkou jsou však imaginární zařízení IMQ a IFB. Zařízení IMQ zmíněno nebude, z důvodu neodladěnosti a v literatuře publikované nestability. Z toho také plyne důvod, proč zařízení IMQ doposud nebylo oficiálně zahrnuto do linuxového jádra. Nicméně stojí za zmínku, že podpora pro IMQ vyžaduje aplikaci patche jádra systému a balíku iproute2.

Zařízení IFB, které má podobné vlastnosti jako IMQ, však bylo do jádra implementováno od verze 2.6.16. Je stabilní a bylo by škoda jej nevyužívat. Nicméně distribuční jádro podporu pro IFB nemá standardně zakompilovanou. Je tedy potřeba kompilovat.

5.1.1 Kompilace jádra

Pro kompilaci jádra je nutné mít k dispozici zdrojové kódy jádra systému a sadu programů určených přímo pro samotnou kompilaci a sestavení jádra. "Velké distribuce" obsahují důmyslné nástroje pro instalaci a konfiguraci softwaru, proč je tedy s výhodou nevyužít. V distribuci Debian je to aplikace apt-get, nebo její grafická nástavba aptitude.

Doporučený způsob kompilace jádra v Debianu vyžaduje balíky: fakeroot, kernel-package, kernel-source a další, které již mohou být v systému přítomny. Úplný seznam je v souboru `/usr/share /doc/kernel-package/README.gz` . Místo balíku kernel-source je možné použít zdrojové kódy tzv. Vanilla jádra, tedy jádra, které neprošlo "distribuční" optimalizací. Výhodou oproti "distribučním" jádrům je rychlost vývoje nových verzí, s čímž se pojí větší kompatibilita s novým hardwarem a podpora nových technologií čerstvě implementovaných do jádra linuxu. Tato možnost bude využita.

Vanilla jádro je k dispozici na serveru www.kernel.org nebo na některém z jeho mirrorů, v České republice je to například server Masarykovy univerzity v Brně ftp://ftp.fi.muni.cz. Jádro řady 2.6 se vyskytuje v adresářové struktuře v adresáři *ftp://ftp.fi.muni.cz/pub/linux/ftp.kernel.org/pub/linux/kernel/v2.6/*.

Práce popisuje kompilaci "nejmladší" verze jádra. V době psaní tohoto dokumentu je to verze 2.6.20. Stažený soubor linux-2.6.20.tar.bz2 je nutné zkopírovat do /usr/src a rozbalit příkazem:

```
tar xjf /usr/src/linux-2.6.20.tar.bz2
```

Z důvodu kompatibility s ostatními programy je vhodné vytvořit symbolický link na nově vzniklý adresář příkazem:

```
ln /usr/src/linux-2.6.20 /usr/src/linux -s
```

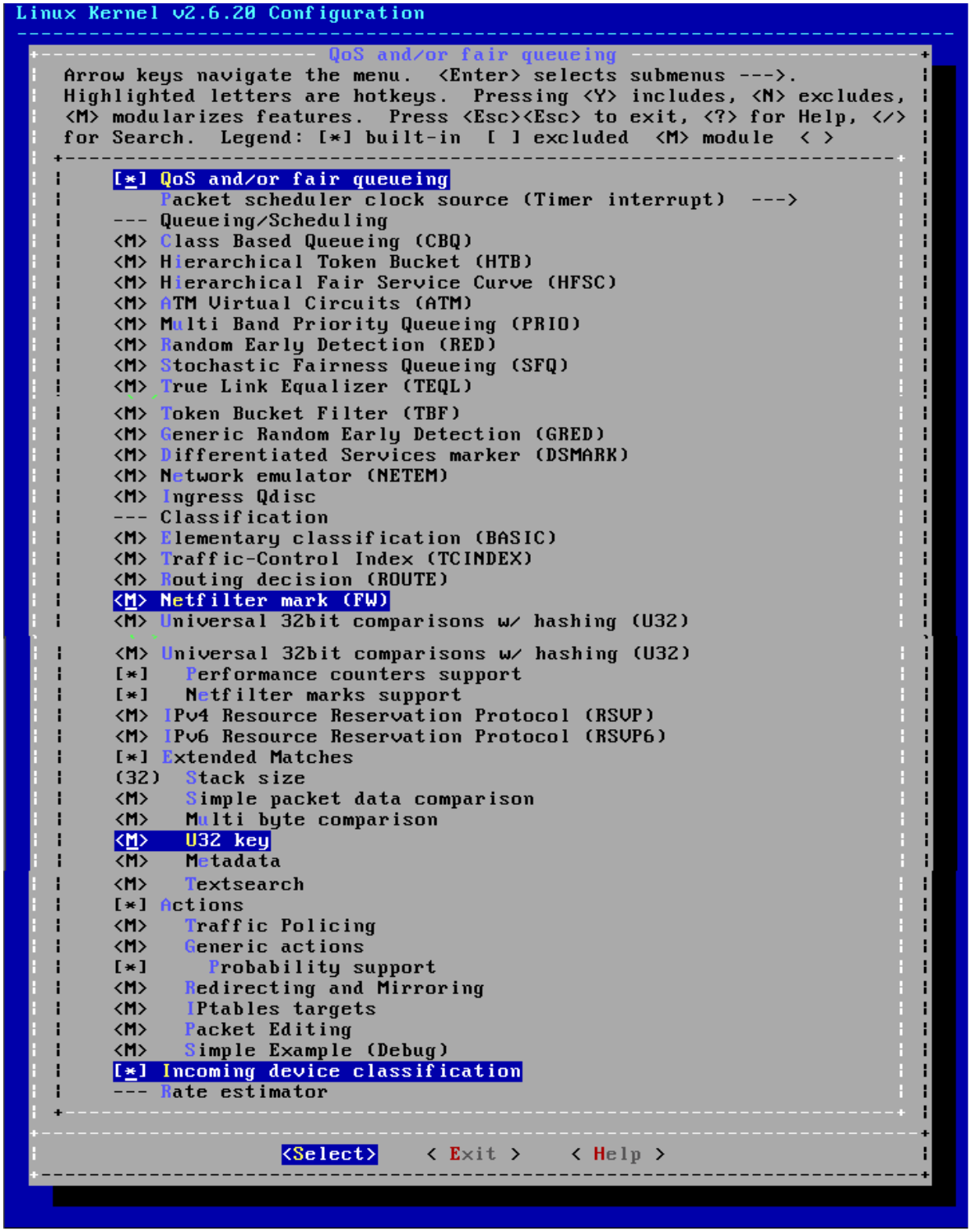
Po přechodu do adresáře se zdrojovými kódy linuxu příkazem:

```
cd /usr/src/linux-2.6.20
```

je nutné spustit skript pro vytvoření konfiguračního souboru pro kompilaci jádra (podmínkou je mít nainstalován balík ncurses-dev):

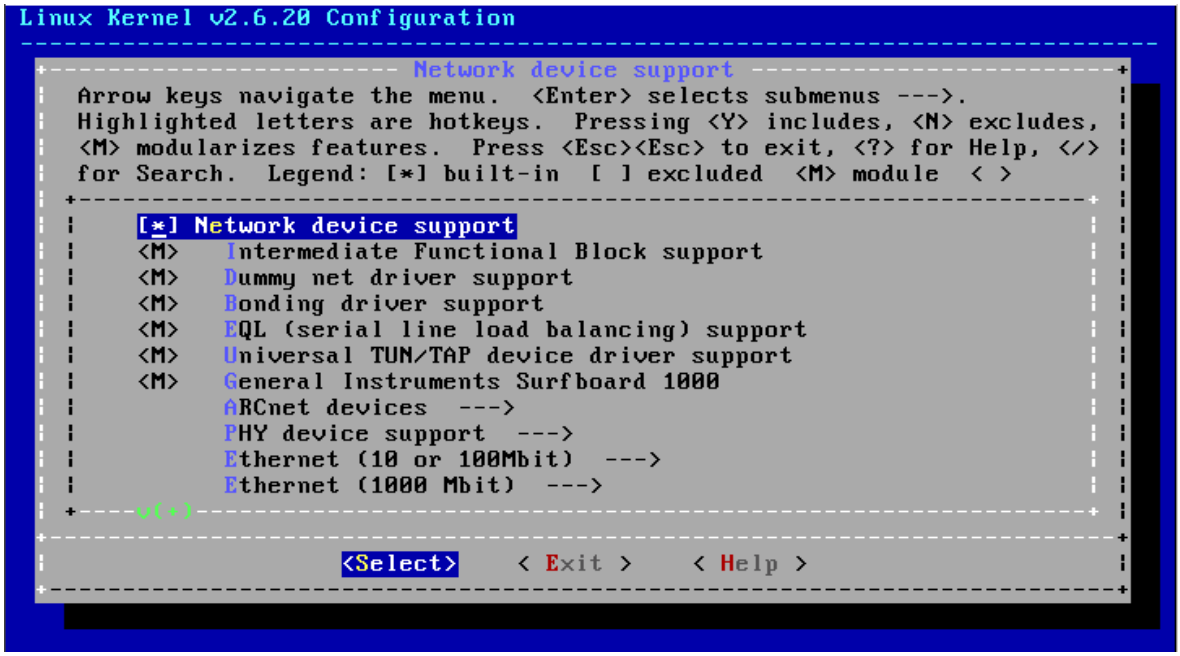
```
make menuconfig
```

V menu v sekci **Networking->Networking options->QoS and/or fair queueing** lze nastavit disciplíny, které bude QoS používat. Viz. obrázek 5.



Obrázek 5 - Konfigurace kernelu

Podpora imaginárního zařízení IFB se zapíná v sekci **Device Drivers->Network device support->Intermediate Funkcional Block support**



Obrázek 6 - Konfigurace kernelu

Po nastavení všech potřebných parametrů lze program ukončit s uložením změn. Následuje příkaz pro pročištění stromu zdrojových textů a vynulování předchozího nastavení balíku kernel-package:

```
make-kpkg clean
```

Nyní nastal čas na samotnou kompilaci jádra. Ta se spouští příkazem:

```
fakeroot make-kpkg --initrd --revision=jadro.1.0 kernel_image
```

Kompilace zabere relativně dost času je závislá na rychlosti počítače, na kterém kompilace probíhá. Po ukončení kompilace se v umístění /usr/src nachází balík

```
linux-image-2.6.20_jadro.1.0_i386.deb
```

Instalace balíku pomocí:

```
dpkg -i linux-image-2.6.20_jadro.1.0_i386.deb
```

Balík s jádrem je dostatečně "chytrý" a provede i nastavení bootloaderu. Nyní stačí systém restartovat. V menu bootloaderu by měla být k dispozici volba pro zavedení nově zkompilovaného jádra s plnou podporou QoS.

5.2 Balík iproute2

Balík iproute2 lze nainstalovat pomocí standardního balíčkovacího systému.

```
apt-get install iproute
```

Balík iproute2 obsahuje sadu programů pro práci se síťovými zařízeními [9]:

ctstat Utilita pro status připojení

ifcfg Shellový skript wrapper pro příkaz **ip**

ifstat Ukazuje statistiky rozhraní,včetně odeslaných a obdržených paketů

ip Hlavní program. Má několik různých funkcí:

ip link ***[device]*** umožňuje zjistit stav zařízení a provedení změn

ip addr povoluje zjistit adresy a jejich vlastnosti, přidat nové adresy, nebo staré smazat

ip neighbor povoluje zjisit sousední vazby a jejich vlastnosti,přidat nové sousedy a staré vymazat

ip rule umožňuje zjistit rourovací pravidla a změnit je

ip route ukazuje routovací tabulku a umožňuje změnit její pravidla

ip tunnel ukazuje IP tunely a jejich vlastnosti a umožňuje změnu tunelů

ip maddr ukazuje adresy multicastu a jejich vlastnosti a umožňuje změnu multicastu

ip mroute umožňuje nastavit,změnit,nebo vymazat multicastové routování

ip monitor umožňuje stále monitorování stavu zařízení,adres a routování

lnstat Nabízí statistiku sítě. Je to mocnější náhrada starého programu **rtstat**

nstat Ukazuje statistiku sítě

routef Komponenta **ip route** pro vyprázdnění routovacích tabulek

routel Komponenta **ip route** pro procházení routovacích tabulek

rtacct Zobrazuje obsah `/proc/net/rt_acct`

rtmon Utilita pro monitorování routování

rtpr Převádí výstup **ip -o** zpět do čitelné podoby

rtstat Utilita zobrazující stav routování

ss Podoné příkazu **netstat**;ukazuje aktivní spojení

tc

Program pro kontrolu provozu sítě(Traffic Controlling); je použit pro implementaci Quality Of Service (QoS) a Class Of Service (CoS)

tc qdisc umožňuje nastavení queueing discipline

tc class umožňuje nastavení tříd založených na plánování queuing discipline

tc estimator umožňuje odhadovat provoz na síti

tc filter umožňuje nastavit filtrování paketů pomocí QoS/CoS

tc policy umožňuje nastavení pravidel QoS/CoS

Za účelem provozování QoS bude dále probrán program **tc.**

5.2.1 Program tc

5.2.1.1 Řazení do front

Základní syntaxe je:

```
tc qdisc [ add | del | replace | change | get ] dev STRING
       [ handle QHANDLE ] [ root | parent CLASSID ]

       [ estimator INTERVAL TIME_CONSTANT ]

       [ [ QDISC_KIND ] [ help | OPTIONS ] ]

        tc qdisc show [ dev STRING ]

kde:
QDISC_KIND := { [p|b]fifo | tbf | prio | cbq | red | etc. }
```

Interpretace jednotlivých částí:

HANDLE reprezentuje jedinečné označení řazení do front, přiděluje jej administrátor. Řazení do front mají druhou složku čísla nulovou (např. 1:0).

ROOT označuje frontu, která je na vrcholu hierarchie tříd a front pro dané zařízení.

PARENT obsahuje odkaz na označení (HANDLE) nadřazené fronty či třídy.

ESTIMATOR je používán pro zjištění, zda byly uspokojeny požadavky fronty.

5.2.1.2 Vytváření tříd

Základní syntaxe je:

```
tc class [ add | del | change | get ] dev STRING

       [ classid CLASSID ] [ root | parent CLASSID ]

       [ [ QDISC_KIND ] [ help | OPTIONS ] ]

        tc class show [ dev STRING ] [ root | parent CLASSID ]

Kde:

QDISC_KIND := { prio | cbq | etc. }
```

Hodnota dosazená za QDISC_KIND musí být fronta, která podporuje třídy jako například HTB a nikoliv tedy TBF.

Popis jednotlivých voleb je následující:

CLASSID je uživatelem přidělené číslo ve formátu *hlavní:vedlejší*. Hlavní číslo je shodné s předkem, vedlejší pak volí administrátor. Například je-li rodič 1:0, může mít třída číslo 1:2.

ROOT značí, že třída je kořenem hierarchie.

PARENT označuje předka v hierarchii.

Třídy v HTB:

Pro vytváření tříd v HTB je k dispozici několik parametrů, kterých lze využít:

```
... qdisc add ... htb [default N] [r2q N]
... class add ... htb rate R1 [burst B1] [mpu B] [overhead 0]
                      [prio P] [slot S] [pslot PS]
                      [ceil R2] [cburst B2] [mtu MTU] [quantum Q]
```

Z toho nejdůležitější jsou:

DEFAULT označuje výchozí frontu. Pokud provoz není zařazen do své fronty, bude přiřazen do výchozí fronty.

RATE rychlost alokovaná pro danou třídu.

PRIO třída s nejnižším parametrem prio je zařazena do fronty jako první.

CEIL rychlost, kterou může třída využít, pokud její nadřazená třída má k dispozici volné pásmo, maximální možná rychlost pro danou třídu.

BURST pokud třída, pro kterou je burst definován využívá šířku pásma pomaleji než má nadefinováno (např. internetové rádio využívá jen 128kbps z 256kbps definovaných pro třídu), HTB si "zapamatuje", kolik bajtů bylo nevyužito. Maximální velikost této "paměti" je právě burst. Pokud je požadavek na přenesení dat nad definovaný limit, HTB umožní přenést až tolik bajtů bez omezení rychlosti, kolik je jich právě "zapamatováno".

CBURST jako burst, ovšem vztahující se k hodnotě ceil.

Ukázková konfigurace HTB je uvedena v příloze PI.

5.2.1.3 Filtry

Základní syntaxe je:

```
tc filter [ add | del | change | get ] dev STRING

      [ pref PRIO ] [ protocol PROTO ]

      [ estimator INTERVAL TIME_CONSTANT ]

      [ root | classid CLASSID ] [ handle FILTERID ]

      [ [ FILTER_TYPE ] [ help | OPTIONS ] ]


tc filter show [ dev STRING ] [ root | parent CLASSID ]

      Kde:

      FILTER_TYPE := { rsvp | u32 | fw | route | etc. }

      FILTERID := ... format depends on classifier, see there

      OPTIONS := ... try tc filter add <desired FILTER_KIND> help
```

Význam jednotlivých voleb je následující:

PREF určuje prioritu filtru.

PROTOCOL určuje protokol paketů, které tento filtr klasifikuje. Žádné dva filtry nemohou mít stejnou prioritu a zároveň protokol.

HANDLE je identifikační číslo filtru

5.2.1.3.1 Filtr route

Filtr route se rozhoduje na základě informací z routovacích tabulek. Značkování paketů se provádí příkazem ip.

```
tc filter ... route [ from REALM | fromif TAG ] [ to REALM ]

            [ flowid CLASSID ] [ police POLICE_SPEC ]
```

REALM určuje značku přidělenou příkazem ip a flowid pak cílovou třídu.

Příklad:

```
#tc filter add dev eth0 parent 1:0 protocol ip prio 100 route to 1
#flowid 1:3
#ip route add 10.0.0.100 via 10.0.0.1 dev eth0 realm 1
```

5.2.1.3.2 Filtr fw

Filtr fw se rozhoduje na základě značek, které do paketů vkládá firewall iptables s volbou

-j MARK --set-mark. S výhodou je možné používat všechny vymoženosti firewallu iptables, značkovat lze provoz podle ip adres, portů, atd.

```
tc filter ... fw [ classid CLASSID ] [ police POLICE_SPEC ]
```

Příklad:

```
#iptables -t mangle -A FORWARD -o eth0 -s 10.0.0.100/32 -j MARK --set-
#mark 21
#tc filter add dev eth0 parent 1:0 protocol ip handle 21 fw flowid 1:3
```

Ukázková konfigurace filtru fw pro třídy v HTB je uvedena v příloze PI.

5.2.1.3.3 Filtr u32

Filtr u32 [1] je nejobecnějšíj klasifikátorem v tomto systému. Cenou za to je oproti předchozím dvěma menší rychlost vyhodnocení. Nicméně vše, co umí filtry fw a route, zvládne i filtr u32. Filtr může klasifiovat pakety na základě zdrojové a cílové IP adresy, zdrojového a cílového portu, protokolu, pole ToS. Obecně může filtr u32 využívat jokoukoliv část všech hlaviček paketu. Konfigurace filtru je poměrně složitá.

```
tc filter ... u32 [ match SELECTOR ... ] [ link HTID ]
[ classid CLASSID ]  [ police POLICE_SPEC ] [ offset OFFSET_SPEC ]
[ ht HTID ] [ hashkey HASHKEY_SPEC ] [ sample SAMPLE ]
                        or u32 divisor DIVISOR
Where: SELECTOR := SAMPLE SAMPLE ...
SAMPLE := { ip | ip6 | udp | tcp | icmp | u{32|16|8} }
SAMPLE_ARGS FILTERID := X:Y:Z
```

Filtr má spoustu voleb, nyní bude probrána specifikace dat, podle kterých se paket klasifikuje.

Položka SAMPLE určuje podtyp filtru u32. Tyto podtypy je možné rozdělit na specifické a obecné. Takže volby ip, ip6,udp, tcp a icmp jsou specifické podtypy (selektory protokolu), které umožňují klasifikaci podle údajů z příslušných hlaviček paketů. Za selektorem hlavičky následuje selektor pole v hlavičce. Tímto je jednoznačně určené místo v hlavičce paketu. Nyní je nutné uvést jen vzor a masku, podle které se budou pakety klasifikovat. V následující tabulce jsou uvedeny možné selektory pole:

selektor protokolu	**selektor pole**	**parametry**	**popis**
match ip	src	prefix/32	zdrojová IP adresa
	dst	prefix/32	cílová IP adresa
	tos		pole TOS
	dsfield	tos u8	pole TOS
	precedence		
	ihl	ihl u8	velikost hlavičky v 32bitových
	protocol	prot u8	slovech
	nofrag		protokol vyšší vrstvy
	firstfrag		paket není fagmentován
	df		příznak první fragment
	mf		příznak "nefragmentovat"
	sport	port u16	příznak "více fagmentů"
	dport	port u16	zdrojový port protokolu vyšší vrstvy
	icmp_type	typ u8	cílový port protokolu vyšší vrstvy
	icmp_code	kod u8	typ zprávy ICMP
			kód zprávy ICMP
match ip6	src	prefix/128	zdrojová IP adresa
	dst	prefix/128	cílový IP adresa
	flowlabel	flow u32	identifikace toku v IPv6
udp, tcp	src	src u16	zdrojový port
	dst	dst u16	cílový port
icmp	type	typ u8	typ zprávy ICMP
	code	kod u8	kód zprávy ICMP

Tabulka 3 - parametry filtru u32

Jsou-li podmínky napsány v jednom volání příkazu za sebou, platí všechny zároveň.

Obecný filtr u32 má syntaxi:

```
tc ... match [ u32 | u16 | u8 ] PATTERN MASK [ at OFFSET | nexthdr+OFFSET]
```

Jedno z klíčových slov u32, u16 nebo u8 specifikuje délku pole v bitech, podle kterého se má klasifikovat. Následovat by měl pattern (vzor) a mask (maska), v délce definované předešlým klíčovým slovem. Offset udává pozici od začátku paketu, nexthdr udává pozici od začátku paketu vyšší vrstvy.

Filtr u32 nabízí opravdu rozmanitá nastavení. Například následující příkazy jsou ekvivalentní:

```
a) tc ... match ip tos 0x10 0xff flowid 1:1
b) tc ... match u8 0x10 0xff at 1 flowid 1:1
```

ad a) Filtru vyhovují všechny pakety, které mají TOS pole nastaveno na 0x10. Pole TOS začíná na druhém bajtu paketu a je jeden bajt velký.

ad b) Filtr pracuje s jednotlivými bajty - to zajišťuje použití použití klíčového slova u8 - 8 bitů = 1 bajt. Konroluje, zda-li je na druhém bajtu paketu nastavena hodnota 0x10 o velikosti jeden bajt. To, že má filtr pracovat s 2. bajtem zajistí parametr at 1. Počítáno od nuly...

Lepšímu pochopení klasifikátoru u32 pomůže pochopení struktury paketu IP [11]:

bity	**0-3**	**4-7**	**8-15**	**16-18**	**19-31**
0	Verze	Délka hlavičky	TOS	Celková velikost paketu	
32	Identifikace			Flagy	Fragment Offset
64	Time to Live		Protokol	Kontrolní součet hlavičky	
96	Zdrojová adresa				
128	Cílová adresa				
160	Nastavení				
160 nebo 192+	Data				

Tabulka 4 - struktura paketu IP

Některé parametry filtru u32 nejsou dokumentovány. Patří k nim parametr link a classid. U jiných parametrů je dokumentace velmi strohá ht - hashovací tabulka, hashkey - klíč hashovací tabulky. V literatuře a také v různých implementacích se tyto parametry vyskytují spíše sporadicky. Výjimku tvoří parametr police.

```
tc ... police rate BPS burst BYTES[/BYTES] [ mtu BYTES[/BYTES] ]

[ peakrate BPS ] [ avrate BPS ] [ ACTION ]

Where: ACTION := reclassify | drop | continue
```

Police slouží k především k jednoduchému omezování příchozího provozu, při použití drop jsou pakety jednoduše zahazovány.

Příklad:

Je požadováno, aby pakety, které přicházejí z ip adresy 192.168.200.200 měly největší možnou rychlost 100 kbit.

```
#tc filter add dev eth0 parent 1: protocol ip prio 10 u32 match ip src
#192.168.200.200/32 flowid 1:2 action police rate 100kbit burst 90k drop
```

Několik příkladů využití filtru u32 [5]:

```
## Výběr pomocí zdrojové adresy
# tc filter add dev wlan0 parent 1: prio 11 protocol ip u32 \
# match ip src 10.107.0.0/16 flowid 1:20


## Výběr pomocí cílové adresy je obdobný
# tc filter add dev wlan0 parent 1: prio 11 protocol ip u32 \
# match ip dst 10.107.0.0/16 flowid 1:20

## Výběr protokolu TCP (6 viz /etc/protocols)
# tc filter add dev wlan0 parent 1: prio 11 protocol ip u32 \
# match ip protocol 6 0xff flowid 1:20

## Výběr podle protokolu pomocí obecného kvalifikátoru u8 (na 9 bajtu IP
## datagramu je číslo protokol vyšší vrstvy). u8 vybere devátý bajt
## provedebitové AND s 0xff a porovná s hodnotou 6.
# tc filter add dev wlan0 parent 1: prio 11 protocol ip u32 \
# match u8 6 0xff at 9 flowid 1:20

## Vybrat TCP ACK pakety menší než 64 - kombinace pravidel. První vybírá
## protokol TCP. Další zajišťuje, že paket nemá volitelné položky záhlaví
## (velikost
## hlavičky ip datagramu ve čtyřbajtech je rovna 5). Třetí zajišťuje, že
## velikost paketu bude menší než 64 a nakonec je test na ACK bit TCP
## paketu.
# tc filter add dev wlan0 parent 1:0 protocol ip prio 10 u32 \
# match ip protocol 6 0xff \
# match u8 0x05 0x0f at 0 \
# match u16 0x0000 0xffc0 at 2 \
# match u8 0x10 0xff at 33 \
# flowid 1:10

## Výběr pomocí zdrojového portu ( protokol TCP; služba ftp-data)
# tc filter add dev wlan0 parent 1: prio 10 protocol ip u32 \
# match ip protocol 6 0xff \
# match ip sport 20 0xffff \
# flowid 1:30
```

```
## Výběr pomocí cílového portu (protokolu UDP; služba DNS)
# tc filter add dev wlan0 parent 1: prio 10 protocol ip u32 \
# match ip protocol 17 0xff \
# match ip dport 53 0xffff \
# flowid 1:10

## Výběr podle ToS pole IP paketu (hodnota Minimize-Delay; například ssh,
## telnet)
# tc filter add dev wlan0 parent 1:0 protocol ip prio 10 u32 \
# match ip tos 0x10 0xff \
# flowid 1:10
```

5.2.1.4 IFB (Intermediate Funkcional Block)

Pokud na routeru běží nějaká služba, u které je požadavek zajistit kvalitu služby pro příchozí provoz (např. upload na lokální ftp server), lze použít pokročilý policing, je možno také říct shaping příchozího provozu, pomocí zařízení IFB. Nevýhodou IFB je nemožnost použití firewallu iptables pro kompletní značkování paketů a to z důvodu implementace IFB do jádra - IFB se váže před netfilter. Je tedy nutné používat tc filtry. Princip fungování IFB je popsán výše. Ukázková konfigurace IFB využívajícího tříd HTB a filtru u32 je uvedena v příloze PII.

Příklad [8]:

```
##Vytvoření fronty htb na zařízení ifb0

#$TC qdisc add dev ifb0 root handle 1:0 htb

##Vytvoření fronty pro příchozí provoz

#$TC qdisc add dev eth0 ingress

##Vytvoření třídy pro přiřazení rychlosti

#$TC class add dev ifb0 parent 1:0 classid 1:10 htb rate 128 kbit

##Vytvoření filtru pro frontu pro příchozí provoz na zařízení eth0,

##zvýrazněná část kódu označí všechny pakety a přesměruje je na ifb0

#$TC filter add dev eth0 parent ffff: protocol ip prio 10 u32 match u32 0
#0 flowid 1:1 action ipt -j MARK --set-mark 1 action mirred egress
#redirect dev ifb0

#Přiřazení rychlosti

#$TC filter add dev ifb0 parent 1:0 protocol ip handle 1 fw classid 1:10
```

5.3 Mastershaper

Mastershaper (http://www.mastershaper.org) je webové rozhraní pro kontrolu a nastavení síťového provozu na routerech a jiných síťových zařízeních, které lze spravovat operačním systémem GNU/Linux. Uživatelé mohou pohodlně definovat pravidla pro chování sítě, mohou také sledovat statistiky síťového provozu na routeru pomocí vydařených grafů, které program mastershaper dynamicky generuje. Pokud je na routeru Mastershaper zprovozněn, nemusí mít administrátor sítě takřka žádné znalosti o operačním systému GNU/Linux, skriptování a jiných nástrojích potřebných pro zprovožnění shapingu a dokáže efektivně regulovat provoz na síti.

Princip fungování Mastershaperu je jednoduchý. Program generuje příkazy pro programy tc a iptables, které jsou posléze předávány jádru. Celý program je napsán ve skriptovacím jazyce php, pro jeho zprovoznění je tedy nutné mít spuštěn nějaký web server s podporou php, např. Apache (http://www.httpd.apache.org). Konfigurace je ukládána do databázových tabulek MySQL (http://www.mysql.com), v síti se tedy musí vyskytovat alespoň jeden server MySQL. Ten nemusí běžet lokálně na routeru. Program také podporuje virtuální zařízení IMQ.

5.3.1 Instalace

Požadavky na systém:

- Linuxové jádro verze 2.4 nebo 2.6
- balík iproute2
- IMQ virtuální zařízení (není podmínkou pro provoz)
- web server s podporou PHP
- PHP4 nebo PHP5 s podporou JPEG, libgd a MySQL
- MySQL databáze
- PHP moduly DB a Net_IPv4
- Perl5 s rozhraním DBD (DBD-MySQL)
- sudo

Instalace serveru

Všechny výše popsané programy a podpůrné knihovny lze nainstalovat pomocí správce balíků softwaru pro Debian - programu aptitude.

Vytvoření nové databáze

Pro zprovoznění Masterhaperu je nutné mít novou databázi. Pro pohodlnější administrování serveru MySQL je vhodné nainstalovat si aplikaci phpMyAdmin (http://www.phpmyadmin.net). Aplikace phpMyAdmin potřebuje pro svůj běh web server s podporou php. Po stažení instalačního balíku phpMyAdmina, který je ve formě zip souboru, stačí obsah balíku rozbalit do nějakého adresáře web serveru, například adresář http://server/admin. V ukázkové instalaci je vytvořena databáze mastershaper a uživatel

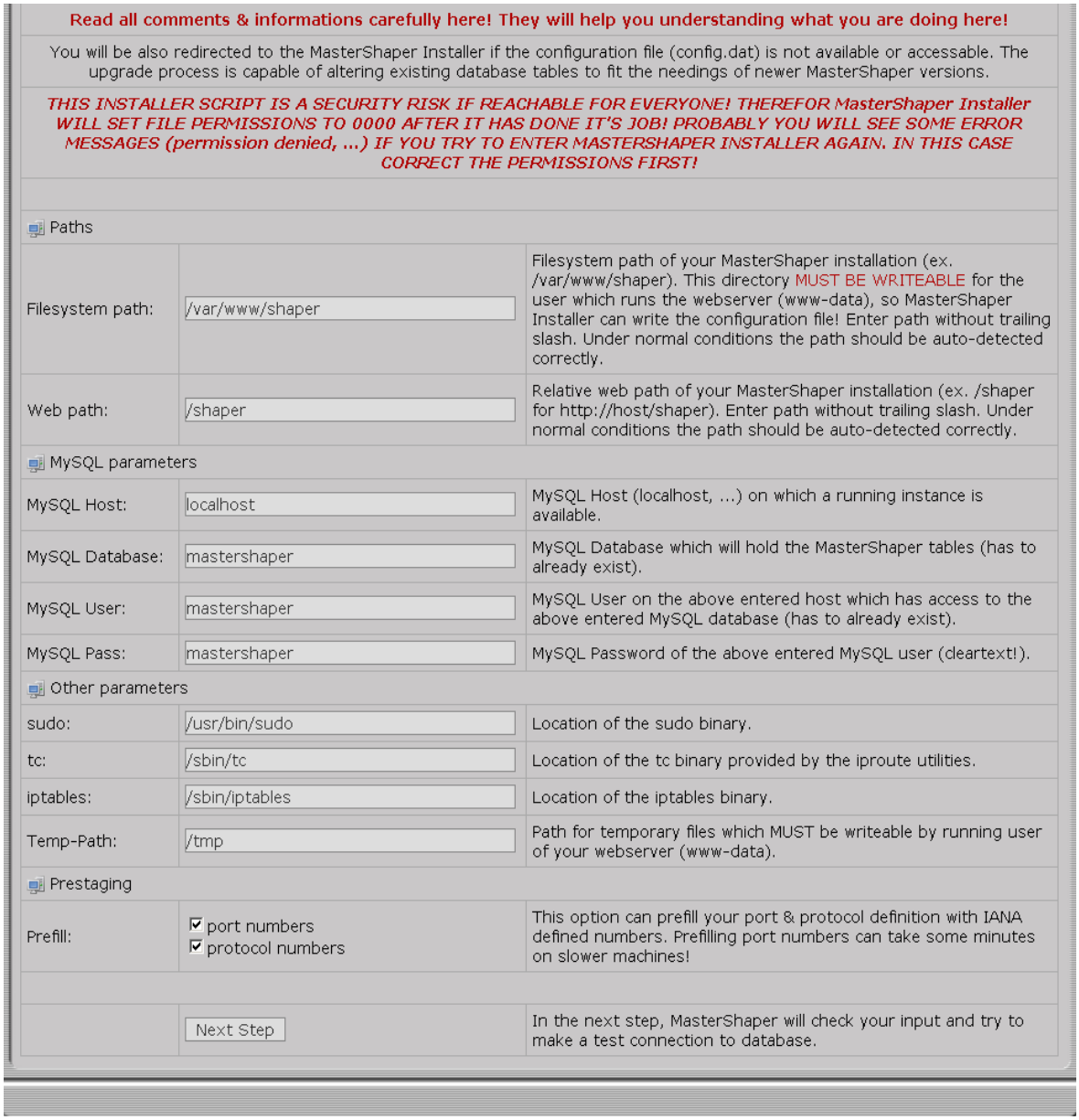
mastershaper mající do ní přístup.

Instalace Mastershaperu

Instalace programu Mastershaper je podobná jako instalace phpMyAdmina. Obsah instalačního balíku je třeba rozbalit do nějakého adresáře na web serveru, například http://server/shaper. Do tohoto adresáře je dále nutné nainstalovat jpgraph (http://www.aditus.nu/jpgraph/), instalace probíhá tak, že se obsah instalačního balíku rozbalí do adresáře http://server/shaper/jpgraph. Podobně se instaluje i phplayersmenu (http://phplayersmenu.sourceforge.net) do http://server/shaper/phplayersmenu.

Instalace PHP-Pear modulů příkazem: *pear install DB Net_IPv4*

Nyní lze přistoupit ke konfiguraci Mastershaperu. Administrace se provádí přes webové rozhraní. Ve webovém prohlížeči na adrese http://server/shaper/setup se vybírají parametry programu. Viz. obrázek 7.



Read all comments & informations carefully here! They will help you understanding what you are doing here!

You will be also redirected to the MasterShaper Installer if the configuration file (config.dat) is not available or accessable. The upgrade process is capable of altering existing database tables to fit the needings of newer MasterShaper versions.

THIS INSTALLER SCRIPT IS A SECURITY RISK IF REACHABLE FOR EVERYONE! THEREFOR MasterShaper Installer WILL SET FILE PERMISSIONS TO 0000 AFTER IT HAS DONE IT'S JOB! PROBABLY YOU WILL SEE SOME ERROR MESSAGES (permission denied, ...) IF YOU TRY TO ENTER MASTERSHAPER INSTALLER AGAIN. IN THIS CASE CORRECT THE PERMISSIONS FIRST!

Paths		
Filesystem path:	/var/www/shaper	Filesystem path of your MasterShaper installation (ex. /var/www/shaper). This directory MUST BE WRITEABLE for the user which runs the webserver (www-data), so MasterShaper Installer can write the configuration file! Enter path without trailing slash. Under normal conditions the path should be auto-detected correctly.
Web path:	/shaper	Relative web path of your MasterShaper installation (ex. /shaper for http://host/shaper). Enter path without trailing slash. Under normal conditions the path should be auto-detected correctly.
MySQL parameters		
MySQL Host:	localhost	MySQL Host (localhost, ...) on which a running instance is available.
MySQL Database:	mastershaper	MySQL Database which will hold the MasterShaper tables (has to already exist).
MySQL User:	mastershaper	MySQL User on the above entered host which has access to the above entered MySQL database (has to already exist).
MySQL Pass:	mastershaper	MySQL Password of the above entered MySQL user (cleartext!).
Other parameters		
sudo:	/usr/bin/sudo	Location of the sudo binary.
tc:	/sbin/tc	Location of the tc binary provided by the iproute utilities.
iptables:	/sbin/iptables	Location of the iptables binary.
Temp-Path:	/tmp	Path for temporary files which MUST be writeable by running user of your webserver (www-data).
Prestaging		
Prefill:	☑ port numbers ☑ protocol numbers	This option can prefill your port & protocol definition with IANA defined numbers. Prefilling port numbers can take some minutes on slower machines!
	Next Step	In the next step, MasterShaper will check your input and try to make a test connection to database.

Obrázek 7 - Instalace programu Mastershaper

Při správném nastavení parametrů vypíše v dalším kroku instalační program zprávu o úspěšném přihlášení k serveru mysql, v dalším kroku zkontroluje zda v databázi existují potřebné tabulky. Pokud ne, instalační program je vytvoří. Po vytvoření tabulek se uživateli zobrazí úvodní stránka. Viz obrázek 8:

Obrázek 8 - Mastershaper - úvodní obrazovka

5.3.2 Konfigurace Mastershaperu

Pro správnou funkci Mastershaperu je důležité jeho nastavení. Nastavení se provádí v menu **Settings**.

Settings->Targets - nastavení cílů pro shaping

Jako cíl lze nastavit ip adresu, skupinu ip adres, nebo fyzickou adresu adaptéru.

Settings->Ports - nastavení portů, na kterých bude možno shapeovat

Settings->Protocols - definice protokolů, které lze použít při shapingu

Settings->Service Levels - vytvoření rychlostních tříd, s možností přiřazení priority

Settings->Options - nastavení programu

Inbound Bandwidth: maximální příchozí rychlost

Outbound Bandwidth: maximální odchozí rychlost

Incoming Interface: příchozí rozhraní

Outgoing Interface: odchozí rozhraní

Classifier: klasifikátor (HTB, HFSC, CBQ)

Default Queuing Discipline: výchozí řazení do front (SFQ, ESFQ, HFSC)

Traffic filter: výběr značkování paketů (pomocí iptables, nebo tc-filter)

Mode: režim (router, bridge)

Settings->Users - definice uživatelů majících přístup k Mastershaperu a definice oprávnění.

Po nastavení v menu settings je potřeba přejít na záložku **Manage**, kde se definují jednotlivá pravidla pro shaping.

Manage->Chains - vytvoření rychlostních tříd pro definované cíle. Přiřadí rychlostní třídu (**Service Levels**) definovanému cíli (**Settings->Targets**)

Manage->Filters - definice filtrů

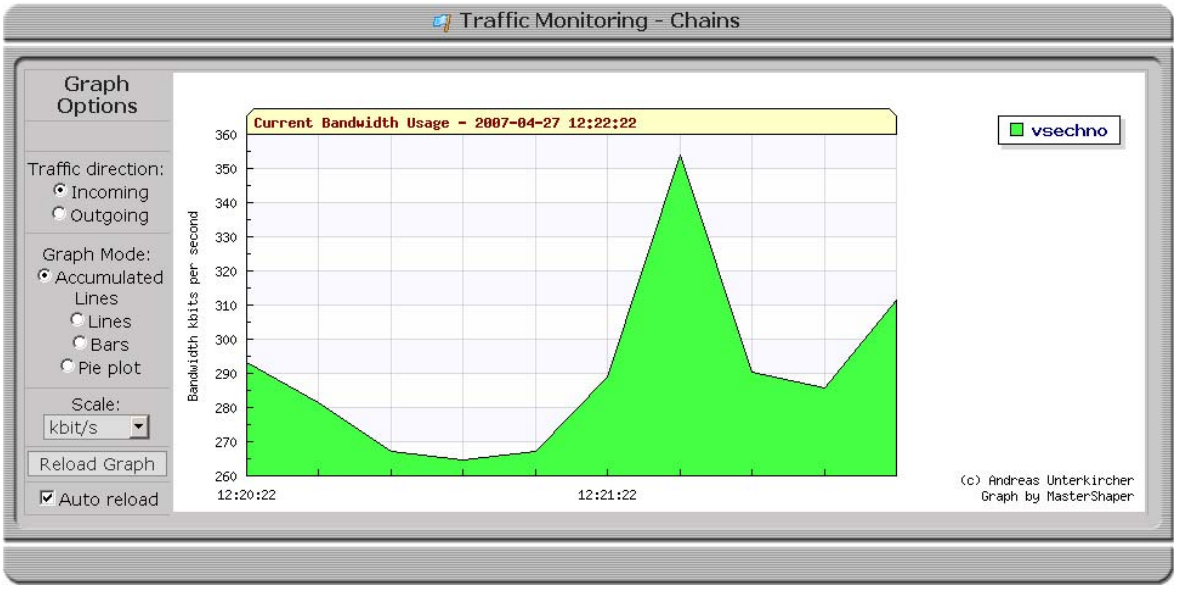
Manage->Pipes - definice pravidel shapingu. Přiřazení definovaného filtru definované rychlostní třídě.

Provoz přes router lze sledovat v menu **Monitoring**. K dispozici jsou tři monitory.

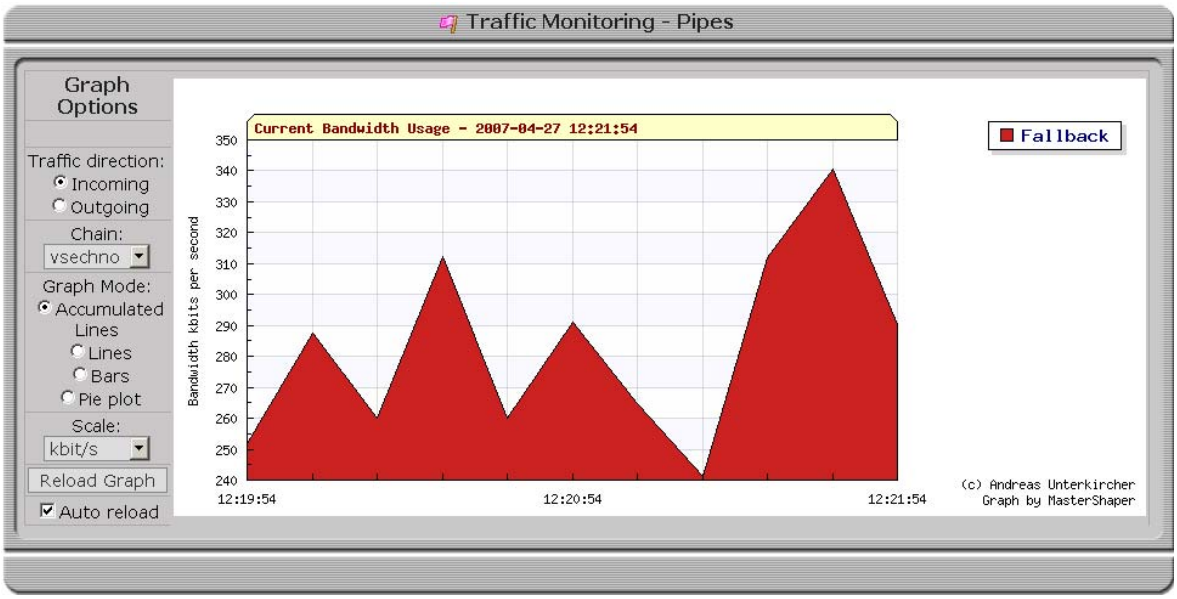
Monitoring->Chains - monitorování rychlostních tříd

Monitoring->Pipes - monitorování jednotlivých pravidel shapingu.

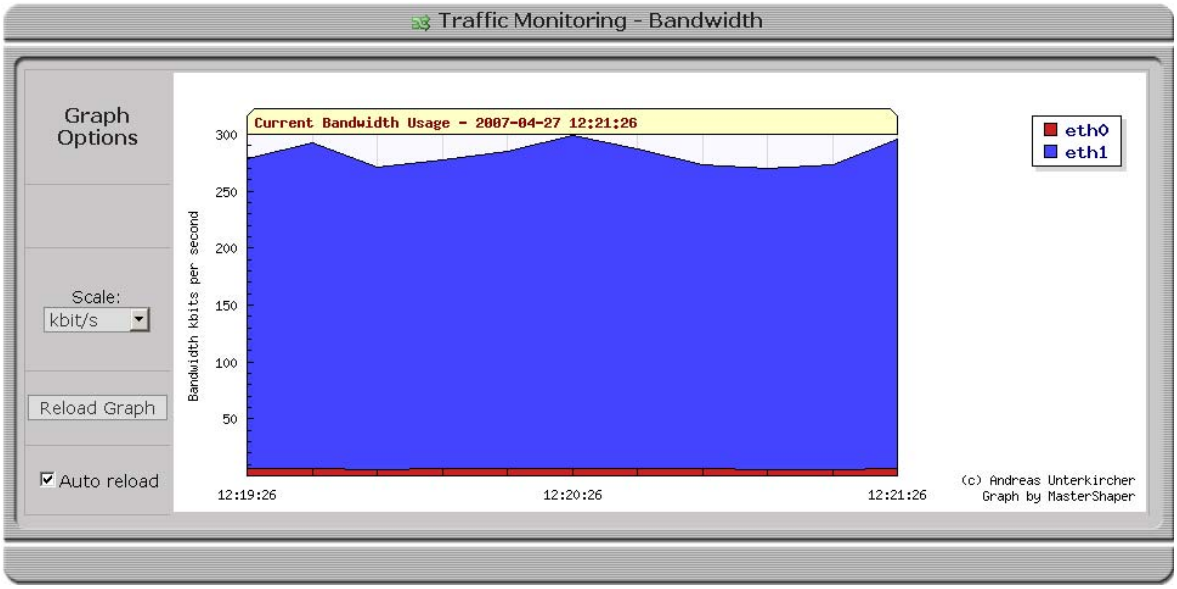
Monitoring->Bandwitch - monitorování rychlosti přenosu dat přes router



Obrázek 9 - Mastershaper kontrola přenosu dat I



Obrázek 10 - Mastershaper - kontrola přenosu dat II



Obrázek 11 - Mastershaper - kontrola přenostu dat III

ZÁVĚR

Práce přináší pro čtenáře podrobný návod na zprovoznění směrovače podporujícího QoS s řadou podrobného nastavení parametrů QoS. Po představení principu kontroly přenosu dat jsou poměrně detailně probrány možnosti programu tc, s velkým množstvím ukázkové konfigurace.

Rozsáhlá část je věnována popisu filtru u32, který nabízí možnosti filtrovat provoz na síti na několika úrovních - od zdrojové/cílové ip adresy, přes filtraci portů, až po analýzu jednotlivých bitů paketu ip.

Pro filtraci příchozí komunikace bylo představeno virtuální zařízení IFB, jež nahrazuje svými vlastnostmi virtuální zařízení IMQ.

Představen byl program MasterShaper, který přináší konfiguraci směrovače pomocí programu `tc` nový rozměr, nicméně jeho provoz je omezen na výkonné směrovače z důvodu závislosti na web serveru s podporou php a serveru MySQL, jejichž provoz vyžaduje výkonnější hardware. Konfigurace filtrování provozu pomocí filtru u32 je u programu MasterShaper pohodlná a přehledná.

Práce ukazuje, jakým mocným nástrojem pro kontrolu síťového provozu je operační systém GNU/Linux, který bývá díky své multiplatformosti nasazován do směrovačů s různorodým hardwarem a uplatněním.

CONCLUSION

This work guides readers through detailed instruction concerning launching of the router with QoS support. After the introduction of the traffic shaping working principle there are several possibilities of the program tc with a great deal of sample configuration described.

A great deal of the work describes the u32 filter, which deals with the possibilities of the traffic control on several levels - source/destination ip address, port filtering, analyzing particular bits of the ip packet, etc.

The virtual device IFB, that replaces with its properties virtual device IMQ, is presented for the incoming communication filtering.

Furthermore, the program MasterShape is described. It brings new extension for the router configuration with the program tc, but its operating is limited to an efficient router due the web server with the php support and the MySQL server dependence. These servers demands more efficient hardware. The MasterShaper configuration is very comfortable and well-arranged.

This work reflects the operating system GNU/Linux as a powerful tool for the traffic shaping control, which is due its popularity for the funkcionality on multiple hardware configurations very often used.

SEZNAM POUŽITÉ LITERATURY

[1] MÁCHA, Jakub. Kontrola síťového provozu. [s.l.], 2000. 42 s. Masarykova Univerzita Fakulta Informatiky, Brno, duben 2000. Vedoucí diplomové práce Mgr. Jan Kasprzak

[2] KACÁLEK, Jan. Modely pro zajištění kvality služeb IP sítí [online]. 2006 [cit. 2007-03-29]. Dostupný z WWW: <http://amarok.cesketelekomunikace.cz/xkacal00/>

[3] LÁVIČKA, Otakar. Klasifikace provozu, shaping, policing [online]. 2005 [cit. 2007-04-01]. Dostupný z WWW: <http://www.fi.muni.cz/~kas/p090/referaty/2005-podzim/st/qos-xlavick1.html>

[4] Quality of Service [online]. 2006 [cit. 2007-04-01]. Dostupný z WWW: <http://en.wikipedia.org/wiki/Quality_of_Service>

[5] QoS - HKFree wiki [online]. 2006 [cit. 2007-04-03]. Dostupný z WWW: <http://charon.hkfree.org/wiki/index.php/QoS>

[6] HOLČÍK, Lukáš. QoS, klasifikace provozu [online]. 2006 [cit. 2007-04-02]. Dostupný z WWW: <http://www.fi.muni.cz/~kas/p090/referaty/2006-jaro/st/qos.html>

[7] DEVAINE, Max. Traffic shaping - 2 (IMQ a úvod do shapingu) [online]. 2006 [cit. 2007-04-05]. Dostupný z WWW: <http://www.abclinuxu.cz/clanky/site/traffic-shaping-2-imq-a-uvod-do-shapingu>

[8] IFB - LinuxNet [online]. 2006 [cit. 2007-04-05]. Dostupný z WWW: <http://linux-net.osdl.org/index.php/IFB>

[9] Linux advanced Routing & Traffic Control HOWTO [online]. 2003. Dostupný z WWW: <http://www.lartc.org/>

[10] MasterShaper [online]. Dostupný z WWW: <http://www.mastershaper.org/index.php/Main_Page>

[11] IP-Paket-Wikipedia [online]. 2007 [cit. 2007-04-06]. Dostupný z WWW: <http://de.wikipedia.org/wiki/IP-Paket>

[12] BIGELOW, Stephen J. Mistrovství v počítačových sítích. [s.l.] : [s.n.],2004. 979 s

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

SEZNAM PŘÍLOH

PŘÍLOHA P I: KONFIGURAČNÍ SKRIPT PRO HTB

Popis funkčnosti: skript je určen pro směrovač, na kterém neběží lokální služba, u níž je vyžadována kontrola rychlosti uploadu na směrovač, a na kterém je spuštěn nat. Pro dvě rozhraní wan/lan platí - příchozí komunikace není nijak rychlostně omezena, omezování funguje na obou rozhraních v odchozím směru. Jelikož je využit filtr fw, který kontroluje tabulku forward, lze zařízení s vnitřní adresou `192.168.100.2` přiřadit rychlosti pro upload/download do/z vnější sítě. Upload na směrovač např. pro server ftp spuštěný na směrovači je neomezován. Důvod je prostý - pakety jsou značkovány na forwardu.

Využití: jednoduchý směrovač s přiřazováním rychlostí.

```
#! /bin/sh
#KONFIGURACNI SKRIPT PRO HTB

#nastaveni rozhrani routeru
wan=eth0
lan=eth1

#mazu vsechny znacky
iptables -t mangle -F

#cisteni vsech qdiscu
tc qdisc del dev $lan root
tc qdisc del dev $wan root
#vytvarim qdisc
tc qdisc add dev $lan root handle 1:0 htb default 1
tc qdisc add dev $wan root handle 1:0 htb default 1
#vytvarim tridy
tc class add dev $lan parent 1:0 classid 1:1 htb rate 100mbit ceil
100mbit
tc class add dev $lan parent 1:1 classid 1:11 htb rate 10mbit ceil 10mbit
tc class add dev $lan parent 1:1 classid 1:12 htb rate 1mbit ceil 2mbit
```

```
tc class add dev $lan parent 1:1 classid 1:13 htb rate 512kbit ceil 1mbit
tc class add dev $lan parent 1:1 classid 1:14 htb rate 256kbit ceil 512kbit
tc class add dev $lan parent 1:1 classid 1:15 htb rate 128kbit ceil 256kbit
tc class add dev $lan parent 1:1 classid 1:16 htb rate 64kbit ceil 64kbit
tc class add dev $wan parent 1:0 classid 1:1 htb rate 100mbit
tc class add dev $wan parent 1:1 classid 1:21 htb rate 10mbit
tc class add dev $wan parent 1:1 classid 1:22 htb rate 1mbit
tc class add dev $wan parent 1:1 classid 1:23 htb rate 512kbit
tc class add dev $wan parent 1:1 classid 1:24 htb rate 256kbit
tc class add dev $wan parent 1:1 classid 1:25 htb rate 128kbit
tc class add dev $wan parent 1:1 classid 1:26 htb rate 64kbit
#vytvarim filtr
#odchozi traffic z routeru do vnitrni site
#pro handle 1 je přiřazena adresa 192.168.100.2
tc filter add dev $lan parent 1:0 protocol ip handle 1 fw flowid 1:15
tc filter add dev $lan parent 1:0 protocol ip handle 2 fw flowid 1:12
#odchozi traffic z routeru do vnejsi site
tc filter add dev $wan parent 1:0 protocol ip handle 21 fw flowid 1:25
tc filter add dev $wan parent 1:0 protocol ip handle 22 fw flowid 1:22
#znackuje pakety ve smeru do vnitrni site pro adresu 192.168.100.2
iptables -t mangle -A FORWARD -o $lan -d 192.168.100.2/32 -j MARK --set-mark 1
#znackuje pakety ve smeru do vnejsi site pro adresu 192.168.100.2
iptables -t mangle -A FORWARD -o $wan -s 192.168.100.2/32 -j MARK --set-mark 21
```

Celý skript má význam: pro adresu 192.168.100.2 nastav pro upload do vnější sítě rychlost 128kbit garantovanou. Pro adresu 192.168.100.2 nastav pro download z vnější sítě rychlost 128kbit garantovanou, pokud bude dostatek zdrojů pro komunikaci nastav 256kbit (hodnota ceil).

PŘÍLOHA P II: KONFIGURAČNÍ SKRIPT PRO IFB

Popis funkčnosti: skript je určen pro směrovač, na kterém běží lokální služba, u níž je vyžadována kontrola rychlosti uploadu na směrovač. Upload z vnitřní sítě je přesměrován na rozhraní ifb0. Pak je této komunikaci přiřazena daná rychlost. Download do vnitřní sítě je provozován na vnitřním rozhraní směrovače, kde není problém nastavit požadovanou rychlost.

Využití: složitější směrovač s přiřazováním rychlostí uplatňujícím se i na komunikaci s lokálně spuštěnými procesy na směrovači.

```
#! /bin/sh


#nastaveni rozhrani routeru
wan=eth0
lan=eth1
#mazu vsechny znacky iptables
iptables -t mangle -F
#kontrola pritomnosti modulu jadra v pameti
if lsmod|grep ifb
then
    echo "Kernel module IFB loaded..."
else
    echo "Loading kernel module IFB..."
    modprobe ifb
    echo "Done..."
fi
#Je ifb aktivni?"
if ifconfig|grep ifb0
then
    echo "Device ifb0 ready..."
else
    echo "Starting device ifb0..."
```

```
    ip link set ifb0 up
    echo "Done..."
fi
#cisteni vsech qdiscu
tc qdisc del dev ifb0 root
tc qdisc del dev $wan ingress
tc qdisc del dev $lan ingress
tc qdisc del dev $lan root
#vytvarim qdisc
tc qdisc add dev ifb0 root handle 1:0 htb default 10
tc qdisc add dev $wan ingress
tc qdisc add dev $lan ingress
tc qdisc add dev $lan root handle 1:0 htb default 10
#vytvarim tridy
tc class add dev ifb0 parent 1:0 classid 1:1 htb rate 100mbit ceil 100mbit
tc class add dev ifb0 parent 1:0 classid 1:10 htb rate 128kbit
tc class add dev ifb0 parent 1:0 classid 1:11 htb rate 256kbit
tc class add dev ifb0 parent 1:0 classid 1:12 htb rate 512kbit
tc class add dev ifb0 parent 1:0 classid 1:13 htb rate 1mbit
tc class add dev ifb0 parent 1:0 classid 1:14 htb rate 2mbit
tc class add dev ifb0 parent 1:0 classid 1:15 htb rate 3mbit
tc class add dev ifb0 parent 1:0 classid 1:16 htb rate 4mbit
tc class add dev ifb0 parent 1:0 classid 1:17 htb rate 8mbit
tc class add dev $lan parent 1:0 classid 1:1 htb rate 100mbit ceil 100mbit
tc class add dev $lan parent 1:0 classid 1:10 htb rate 128kbit
tc class add dev $lan parent 1:0 classid 1:11 htb rate 256kbit
tc class add dev $lan parent 1:0 classid 1:12 htb rate 512kbit
tc class add dev $lan parent 1:0 classid 1:13 htb rate 1mbit
tc class add dev $lan parent 1:0 classid 1:14 htb rate 2mbit
tc class add dev $lan parent 1:0 classid 1:15 htb rate 3mbit
```

```
tc class add dev $lan parent 1:0 classid 1:16 htb rate 4mbit

tc class add dev $lan parent 1:0 classid 1:17 htb rate 8mbit


#vytvarim filtr

tc filter add dev ifb0 parent 1:0 protocol ip handle 1 fw classid 1:12

#upload z vnitrni site - přesměrování na ifb

tc filter add dev $lan parent ffff: protocol ip prio 10 u32 match ip src
192.168.100.2/32 flowid 1:0 action ipt -j MARK --set-mark 1 action mirred
egress redirect dev ifb0

#download do vnitrni site

tc filter add dev $lan parent 1:0 protocol ip u32 match ip dst
192.168.100.2/32 classid 1:13
```

Celý skript má význam: pro adresu 192.168.100.2 nastav pro upload na směrovač rychlost 512kbit garantovanou. Pro adresu 192.168.100.2 nastav pro download ze směrovače rychlost 1mbit garantovanou.