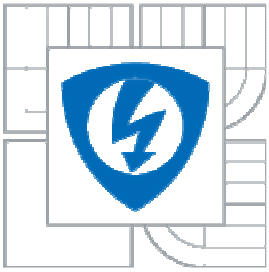
VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA ELEKTROTECHNIKY A KOMUNIKAČNÍCH TECHNOLOGIÍ

ÚSTAV TELEKOMUNIKACÍ

FACULTY OF ELECTRICAL ENGINEERING AND COMMUNICATION

DEPARTMENT OF TELECOMMUNICATIONS

REALIZACE QOS FILTRACE NA IPV6 PROTOKOLU S VYUŽITÍM LINUXU

ESTABLISHMENT OF QOS FILTERING ON IPV6 PROTOCOL USING LINUX

DIPLOMOVÁ PRÁCE

MASTER´S THESIS

AUTOR PRÁCE **BC. FRANTIŠEK MATUŠINA**
AUTHOR

VEDOUCÍ PRÁCE **ING. TOMÁŠ MATOCHA**
SUPERVISOR

BRNO 2010

ANOTACE
Diplomová práce se zabývá implementací QoS filtrace na IPv6 protokolu. Náplní práce je popis protokolu IPv4 a popis protokolu IPv6 a jejich vzájemné porovnání. Celý problém je implementován pod Linux. Z toho důvodu se v práci také věnuji lehkému popisu Linuxových distribucí. V dalších kapitolách popisuji QoS a QoS na vrstvách TCP/IP modelu.
V praktické části je navrženo řešení sítě pracující na protokolu IPv6 po Linuxem. Realizace zahrnuje instalaci serveru, klientů a jejich konfiguraci. Dále je zde obsaženo počítání provozu, řízení provozu a výsledky takto ovlivněných provozů jsou zaznamenány v grafech. Grafy jsou pro jednotlivá měření okomentována a vysvětleny výsledky měření. V přílohách jsou přiloženy veškeré grafy získané při měření a jednotlivé skripty, které byli použity.

Klíčová slova: IPv4, IPv6, Linux, QoS, HTB, ipaccounting, rrd, grafy provozu

ABSTRACT
This diploma thesis is interested in establishment of QoS filtering on IPv6 protocol usány Linux. Thesis describe protocol IPv4 and protocol IPv6 and their differences. Problematic is implement to Linux. Because this problem is implement in Linux, I describe a few distribution of Linux in thesis. Another part of thesis are about QoS and about QoS implement on TCP/IP model.
In practise part is make a propsal resolution working site under protocol IPv6 in Linux. Realize include install of server, clients and their configuration. Realize include script, that account, take control of the traffic. Changeing traffic is show in graphs. Graphs of measure are comented and outcome clear up. All graphs and scripts are add to attachments.

Keywords: IPv4, IPv6, Linux, QoS, HTB, ipaccounting, rrd, graphs of traffic

Prohlášení

Prohlašuji, že svou diplomovou práci na téma „Realizace QoS filtrace na IPv6 protokolu s využitím Linuxu“ jsem vypracoval samostatně pod vedením vedoucího diplomové práce a s použitím odborné literatury a dalších informačních zdrojů, které jsou všechny citovány v práci a uvedeny v seznamu literatury na konci práce.

Jako autor uvedené diplomové práce dále prohlašuji, že v souvislosti s vytvořením této diplomové práce jsem neporušil autorská práva třetích osob, zejména jsem nezasáhl nedovoleným způsobem do cizích autorských práv osobnostních a jsem si plně vědom následků porušení ustanovení § 11 a následujících autorského zákona č. 121/2000 Sb., včetně možných trestněprávních důsledků vyplývajících z ustanovení § 152 trestního zákona č. 140/1961 Sb.

V Brně dne …… podpis autora.....

Poděkování

Děkuji vedoucímu práce Ing. Tomáši Matochovi za velmi užitečnou metodickou pomoc a cenné rady při zpracování diplomové práce.
Dále děkuji panu Ing. Michalovi Fuchsovi za umožnění přístupu do laboratoře počítačové techniky na UREL FEKT v Brně.

V Brně dne …….. podpis autora .....

OBSAH

BRNO UNIVERSITY OF TECHNOLOGY ..... - 1 -
1.Úvod ..... - 8 -
2.Linux ..... - 9 -
2.1 Caldera Network Desktop, OpenLinux ..... - 10 -
2.2 Debian GNU/Linux ..... - 10 -
2.3 Linux Mandrake ..... - 10 -
2.4 RedHat Linux ..... - 10 -
2.5 Slackware Linux ..... - 10 -
2.6 SuSE Linux ..... - 11 -
2.7 Turbo Linux ..... - 11 -
3. IPv4 ..... - 12 -
4. IPv6 ..... - 14 -
4.1 Adresní prostor ..... - 14 -
4.2 Bezstavová konfigurace adres ..... - 15 -
4.3 Multicast ..... - 15 -
4.4 Adresy místní linky ..... - 16 -
4.5 Jumbogramy ..... - 16 -
4.6 Bezpečnost v síťové vrstvě ..... - 16 -
4.7 Mobilita ..... - 16 -
4.8 Nepřítomnost kontrolního součtu ..... - 17 -
4.9 Adresování ..... - 17 -
5. Duality of Service QOS ..... - 18 -
5.1 Důvody sledování provozu na síti ..... - 19 -
5.2 Úrovně QoS ..... - 19 -
5.3 QoS na linkové vrstvě ..... - 19 -
5.4 QoS z hlediska End-to-End konektivity ..... - 20 -
6. QoS ve vrstvách TCP/IP modelu ..... - 20 -
6.1 Historie ..... - 20 -
6.2 Modely pro IP Qos ..... - 21 -
6.2.1 Integrated services – IntServ model ..... - 21 -
6.2.2 Differentiated Services - DiffServ ..... - 22 -
6.3 Klasifikace paketů ..... - 22 -
6.4 Zpracování paketů ..... - 24 -
6.4.1 Expedited forwarding (EF) ..... - 25 -
6.4.2 Assured forwarding (AF) ..... - 25 -
6.4.3 Traffic conditioning ..... - 25 -
6.5 Řízení front ..... - 27 -
6.5.1 Priority queueing (PQ) ..... - 27 -
6.5.2 Class-based queueing ..... - 28 -
6.5.3 Weighted fair queueing (WFQ) ..... - 28 -
6.5.4 (Weighted-) Random Early Detection (RED / WRED) ..... - 29 -
7. Praktická část ..... - 30 -
7.1 Konfigurace serveru ..... - 31 -
7.1.1. Instalace distribuce Linux Arch pro server i klienta ..... - 31 -
7.1.2 Získání adresy IPv6 ..... - 33 -
7.1.3 Konfigurace serveru ..... - 35 -
7.1.3.1 FTP6 ..... - 37 -

7.1.3.2 Ipaccounting a rddht ..... - 39 -
7.1.3.3 HTB ..... - 39 -
7.1.4 Konfigurace klienta ..... - 40 -
7.1.4.1 Připojení klienta na ftp6 ..... - 42 -
7.2 Testování QoS ..... - 43 -
7.2.1 Jeden klient ..... - 43 -
7.2.2 Dva klienti ..... - 45 -
7.2.3 Čtyři klienti při hodnotě rate 1000 kbit/s ..... - 46 -
7.2.4 Čtyři a méně klientů při hodnotě rate 256 kbit/s ..... - 48 -
8. Závěr ..... - 51 -
Literatura ..... - 52 -
Přílohy ..... - 54 -

Seznam obrázků

Obr.1: Návrh sítě hvězdicové topologie……...30
Obr.2: Povolení forwardingu…….35
Obr.3: Příkaz ifconfig……....36
Obr.4.: Příkaz ping6……37
Obr.5: Pure-ftp.conf……38
Obr.6: Ipaccounting……39
Obr.7: Ping6 od klienta na server……...40
Obr.8: Příkaz traceroute6……41
Obr.9: FTP6 – příhlášený klient…….....42
Obr.10: Stahování souboru z ftp6.....42
Obr.11: Stahování souboru jedním klientem……...43
Obr.12: Graf stažení souboru jedním klientem……44
Obr.13: Graf stahování souboru dvěmi klienty současně – klient Fedora……...45
Obr.14: Graf porovnání stahování jedním a dvěmi klienty.……46
Obr.15: Graf stahování čtyř klientů současně při hodnotě rate 1000 kbit/s.……...47
Obr.16: Graf porovnání stahování klienta při různých hodnotách rate……47
Obr.17: Stahování souboru čtyřmi klienty současně……48
Obr.18: Graf klienta při rate 1000 kbit/s a 256 kbit/s při stahování čtyřmi klienty……..49
Obr.19: Graf klienta1 při rate 1000 kbit/s a 256 kbit/s při stahování čtyřmi klienty……49

1.Úvod

Tato diplomová práce se bude zabývat současnými možnostmi QoS na verzi IPv6 protokolu. Daná problematika bude implementována do operačního systému Linux.

Cílem diplomové práce je zevrubné seznámení se s teoretickými předpoklady pro danou problematiku a uvedení získaných poznatků do praktického využití. V první kapitole se teoreticky seznámíme s jednotlivými distribucemi Linux. Jelikož je spousta distribucí a modifikací, seznámíme se s těmi nejzákladnějšími. Ke každé distribuci bude uvedena její specifikace.

Ve druhé kapitole jsou zevrubné teoretické předpoklady pro seznámení se s problematikou protokolu verze 4 a v následující kapitole seznámení s verzí protokolu 6. V kapitole popisující verzi protokolu 6 jsou uvedené pro srovnání i nějaké poznatky o rozdílech s protokolem předešlé verze 4.

Následující kapitoly se zabývají teoretickými podklady o QoS a řízení služeb v DiffServ. Tyto kapitoly se podrobně věnují jednotlivému řízení front, zpracováním a klasifikaci paketů.

Teoretických poznatků získaných v kapitolách o teorii QoS na IPv6 a pomocí poznatků o Linux sestavíme praktickou část, kde vytvoříme síť na protokolu IPv6 a implementujeme zde filtry pro sledování provozu.

2.Linux

Na počátku devadesátých let došlo k velkému rozšíření internetu mezi běžné uživatele a byla zvýšená poptávka po kvalitním operačním systému. Drahý operační systém Unix se úspěšně podařilo modifikovat, aby byly splněny požadavky operačního systému, který bude zdarma a nejlépe i pod veřejnou licencí GPL (GNU General Public Licence). Danou podmínku se podařilo splnit a vznikl operační systém Linux a méně známý a úspěšný GNU/Hurd.

Na vývoji Linuxu se nepodíleli společnosti, které vytvořili původní Unix. Autorem operačního systému Linux je Linus Torvalds, který v té době studoval v Helsinkách informatiku. V jazyce C začal psát vlastní kernel (jádro) nového operačního systému. Ten byl optimalizován pro mikroprocesory Intel 80386. Výchozím operačním systémem byl pro Linuse operační systém Minix. Pojmenováni Linux vychází z křestního jména zakladatele, jen bylo proměněno písmeno s za x.

Verze Linuxu 0.01 byla spustitelná pouze z operačního systému Minix a obsahovala jen holé minimum kernelu. Za první oficiální verzi byl označen Linux 0.02, který již disponoval BASHem a GNU kompilátorem GCC.

Vývoj operačního systému Linux proběhl naprosto jedinečně. Při tvorbě softwaru nebylo použito klasického postupu, kdy je zadán cíl a programátoři, většinou zaměstnanci firmy, se snaží o jeho dosažení. A na základě výsledků je pak zpracována analýza a vyčísleny náklady. Danému případnému prodražování a vysokým nákladům na vývoj a výzkum Linus Tovarlds předešel tím, že se rozhodl podělit se s odbornou veřejností o svůj kernel. Tak vznikl jedinečný projekt, který s mírnými obměnami funguje dodnes. Tímto výrobním procesem je zajištěna vysoká transparentnost a funkčnost vyvíjených systémů.

Díky rozmanitosti a počtu lidí zainteresovaných ve výzkumu, je rozšířeno několik distribucí linuxu:

2.1 Caldera Network Desktop, OpenLinux

Jedná se o komerční distribuci, která mimo jiné obsahuje prostředky pro administraci systému pomocí X11, komerční X Server a další.

2.2 Debian GNU/Linux

Na této distribuci se podílejí lidé z celého světa. Jedná se o nekomerční distribuci Linuxu. Na této distribuci je nejzajímavější její systém pro instalaci a údržbu balíků softwaru – dpkg. Systém umožňuje kompletní definici závislostí mezi jednotlivými balíky.

2.3 Linux Mandrake

Jedná se o poměrně nový typ distribuce. Její základ byl založen na distribuci RedHat. Z distribuce Mandrake využívá z distribuce RedHat stejný systém balíčků (RPM).

2.4 RedHat Linux

Jedná se o nejrozšířenější distribuci v České republice. Distribuce se vyznačuje uživatelsky příjemnou instalační procedurou. Program na konfiguraci systému funguje na bázi X-Window. Prostředky pro administraci systému fungují přes X11. Daná distribuce podporuje velké množství softwarových balíků. Ke správě balíků softwaru a pro instalaci používá systém RPM. Distribuce je volně šiřitelná. Pokud dojde k upgrade systému, není nutno přeinstalovávat celý systém a je možno jej během upgrade běžet. RedHat disponuje i komerční verzí.

2.5 Slackware Linux

V minulosti se jednalo o jednu z nejrozšířenějších distribucí. Výhodou této distribuce je velké množství dostupných balíčků. Dále rychlá instalace, velká rozšířenost, možnost instalace

z pásky, instalace na FAT oblast. Nevýhodou je nemožnost upgrade na vyšší verzi bez přeinstalování systému.

2.6 SuSE Linux

SuSE Linux se dodává v několika jazykových verzích, včetně české. Jedná se o placenou distribuci, která zahrnuje českou dokumentaci i instalační podporu. Pro správu balíčků využívá program RPM. Distribuce poskytuje komfortní konfigurační nástroje pro správu systému, nastavení hardwaru a konfiguraci grafického prostředí.

2.7 Turbo Linux

Jedná se o komerční verzi, která je produkována firmou Turbolinux. Distribuce Turbolinux je velmi rozšířena v asijských zemích. Díky řešení Turbolinux Cluster Server, lze počítače s operačními systémy Linux,Solária a Windows NT kombinovat do výkonných clusterů.

Výše uvedené distribuce Linuxu jsou pouze výběrem nevíce používaných. Výčet zdaleka neobsahuje všechny distribuce a mutace původního kornelu.

3. IPv4

Internet Protocol version 4 (IPv4) je v informatice čtvrtá revize IP (Internet Protocol) a zároveň jeho první verze, která se masivně rozšířila. Spolu s IPv6 vytvářejí základ pro komunikaci v rámci sítě Internet. IPv4 je popsána IETF v RFC 791 (září 1981), které nahradilo RFC 760 (leden 1980) a je standardizována jako MIL-STD-1777 Ministerstvem obrany USA.

IPv4 je datově orientovaný protokol, který je používán v sítích s přepojováním paketů (např. Etnernet). Jde o protokol přepravující data bez záruky, tj. negarantuje ani doručení ani zachování pořadí ani vyloučení duplicit. Zajištění těchto záruk je ponecháno na vyšší vrstvě, kterou v představuje protokol TCP. Stejně tak je na vyšší vrstvě ponechána kontrola integrity dat, protože IPv4 datagram nese pouze informaci o kontrolním součtu hlavičky datagramu se služebními údaji.

Datagram IPv4 obsahuje hlavičku se služebními údaji nutnými pro přepravu a za ní následují data. Konec hlavičky je zarovnán na násobek čtveřice bajtů pomocí výplně (anglicky *padding*). Strukturu IP datagramu vystihuje tabulka uvedená v pravé části. Následuje popis jednotlivých polí:

- **Verze**: verze IPv4
- **Délka hl.**: délka hlavičky ve čtyřbajtových slovech; hlavička může být kvůli volbám různě dlouhá.
- **Typ služby (TOS, Type of Service)**: podle původních představ měla tato položka umožnit odesilateli, aby zvolil charakter přepravní služby ideální pro dotyčný datagram. Jednotlivé bity znamenaly např. požadavek na nejmenší zpoždění, největší šířku pásma či nejlevnější dopravu. Směrování pak mělo brát ohled na hodnotu TOS a volit z alternativních tras tu, která nejlépe odpovídala požadavkům datagramu. V praxi však k realizaci nedošlo. V současnosti se položka používá k podobným účelům – nese značku pro mechanismy zajišťující služby s definovanou kvalitou (QoS).
- **Celková délka**: délka datagramu v bajtech.

- **Identifikace**: odesilatel přidělí každému odeslanému paketu jednoznačný identifikátor. Pokud byl datagram při přepravě fragmentován, pozná se podle této položky, které fragmenty patří k sobě (mají stejný identifikátor).
- **Příznaky**: slouží pro řízení fragmentace. Definovány jsou dva: *More fragments* ve významu „nejsem poslední, za mnou následuje další fragment původního datagramu“ a *Don’t fragment* zakazující tento datagram fragmentovat.
- **Offset fragmentu**: udává, na jaké pozici v původním datagramu začíná tento fragment. Jednotkou je osm bajtů.
- **TTL (Time To Live)**: představuje ochranu proti zacyklení. Každý směrovač zmenší tuto hodnotu o jedničku (případně o počet sekund, které datagram ve směrovači strávil, pokud zde čeká déle). Pokud tím TTL nabude hodnotu nula, datagram zahodí, protože vypršela jeho životnost.
- **Protokol**: určuje, kterému protokolu vyšší vrstvy se mají data předat při doručení. Čísla protokolů definována v RFC 1700 (TCP: 6, UDP: 17, ICMP: 1, EGP: 8, …).
- **Kontrolní součet hlavičky**: slouží k ověření, zda nedošlo k poškození. Počítá se pouze z hlavičky a pokud nesouhlasí, datagram bude zahozen.
- **Adresa odesílatele**: IPv4 adresa síťového rozhraní, které datagram vyslalo.
- **Adresa cíle**: IP adresa síťového rozhraní, kterému je datagram určen.
- **Volby**: různé rozšiřující informace či požadavky. Například lze předepsat sérii adres, kterými má datagram projít. Volby obvykle nejsou v datagramu použity (v tabulce jsou barevně odlišeny).
- **Výplň**: nenese žádnou informaci, slouží k zaokrouhlení délky hlavičky na násobek čtyř bajtů, pokud jsou použity volby uvedené výše.
- **Data**: obsahuje přepravovaná data.

[9]

4. IPv6

Internetový protokol verze 6 je síťová vrstva pro mezisíťový přenos paketů. Bývá označován za následníka internetového protokolu verze 4. Což je současná verze internetového protokolu používaná v internetu.

Hlavním přínosem oproti verzi IPv4 je, že protokol verze 6 nabízí daleko větší adresní prostor. Tato skutečnost umožňuje větší pružnost při přiřazování adres. Prodloužením délky adresy odpadla nutnost použití překladu síťových adres z důvodu vyčerpání adresního prostoru. Také zjednodušuje otázku přidělování adres a přečíslování při změně poskytovatele připojení. Záměrem tvůrců nebylo přiřadit každému počítači jedinečnou pevnou adresu.

Protokol verze 6 obsahuje celkem 2^128 adres. Pokud by jsme tedy vypočítali, kolik připadá adres na jednoho člověka z dnešních 6,5 miliardy, tak se dostaneme na číslo 5x10^28 adres na osobu.

Vysoký počet adres umožňuje hierarchické uspořádání, což zjednodušuje směrování a přečíslování. Pro protokol IPv4 byly vyvinuty mnohdy složité techniky CIDR pro co nejlepší využití omezeného adresního prostoru. Přečíslování při změně poskytovatele připojení může být velmi obtížné. Nicméně, s IPv6 se přečíslování stává téměř automatickým, jelikož identifikace hostů jsou odebrány z identifikátoru poskytovatele připojení. Existují oddělené adresní prostory pro poskytovatele připojení a pro klienty - ty jsou „nedostatečné“ v bitech adresního prostoru, ale velmi efektivní pro provozní záležitosti, jako například změna poskytovatele připojení.

4.1 Adresní prostor

Hlavním důvodem zavedení IPv6 je větší adresní prostor. Adresy IPv6 jsou dlouhé 128 bitů, zatímco adresy u IPv4 byly 32 bitové. Díky většímu adresnímu prostoru odpadá potenciální vyčerpání adresního prostoru IPv4, bez potřeby překladu síťových adres a jiných postupů porušujících přirozenost internetových přenosů jako komunikace mezi dvěma koncovými

body. Překlad síťových adres může být ve výjimečných připadech stále zapotřebí, ovšem návrháři Internetu vědí že bude v IPv6 obtížně realizovatelný, proto se snaží překladu síťových adres vyhnout, kdykoliv je to možné. Také se zjednodušuje správa středních až velkých sítí, díky nepotřebnosti složitých schémat podsítí. Tvorba podsítí se v ideálním případě změní v logické rozdělení IP sítě pro optimální přístup a směrování. Nevýhoda výměnou za rozšířený adresní prostor je jisté zvýšení režijních přenosů oproti IPv4, což může nepříznivě ovlivnit oblasti s omezenou šířkou přenosového pásma (lze využít komprese hlavičky paketů ke zmírnění tohoto problému). Také zapamatování adresy IPv6 může být pro člověka velmi obtížné až nemožné, kvůli její délce. Je proto nutné využívat DNS.

4.2 Bezstavová konfigurace adres

Host v IPv6 může být konfigurován automaticky, pokud je připojen na směrovanou IPv6 síť, za použití zpráv směrem k ICMP v6 serveru. Při prvním připojení k síti host vyšle 'router solicitation multicast žádost o konfigurační parametry na místní linku. Odpovídajícím způsobem nastavený ICMPv6 směrovač odpoví na tuto žádost paketem 'router advertisement' který obsahuje konfigurační parametry síťové vrstvy. Pokud není IPv6 autokonfigurace použitelná, host může využít stavové konfigurace (DHCP v6) nebo být nastaven ručně či jiným způsobem.

4.3 Multicast

Multicast je součástí základní specifikace IPv6 na rozdíl od IPv4, kde byl zaveden později. IPv6 nepoužívá broadcast na místní linku, stejného výsledku je možno dosáhnout pomocí multicastu skupině *all-hosts* (ff02::1). Většina prostředí nicméně v současné době nemá infrastrukturu sítě připravenou na směrování multicastu. Multicast v jednotlivé podsíti bude fungovat, ale v globální nemusí.

4.4 Adresy místní linky

Rozhraní IPv6 má kromě globálních adres často využívaných aplikacemi také adresy místní linky. Ty jsou vždy k dispozici a nikdy se nemění, což zjednodušuje vývoj konfiguračních a směrovacích protokolů.

4.5 Jumbogramy

Protokol IPv4 má omezenou velikost paketu na 64KiB. Naproti tomu IPv6, pokud je použit u komunikace hostů a komunikační cesty umožňující maximální velikost transportní jednotky větší než 65576 oktetů (65536 + 40 na hlavičku), disponuje volitelně podporu pro pakety přes tento limit. Ty se označují jako jumbogramy a jejich velikost může být až 4GiB. Použití jumbogramů může zlepšit výkon odpovídajících sítí.

4.6 Bezpečnost v síťové vrstvě

Protokol pro IP vrstvu šifrování a autentizaci IPsec je integrální součástí souboru protokolů IPv6, na rozdíl od IPv4, kde je přítomen volitelně (obvykle ale implementován). V současnosti však IPsec není využíván, vyjma zabezpečení spojení mezi směrovači BGP.

4.7 Mobilita

Na rozdíl od IPv4, Mobilní IPv6 (MIPv6) překonává trojstranné směrování, a je proto stejně efektivní jako IPv6. Tato výhoda je víceméně hypotetická, jelikož v současné době nejsou v provozu ani MIPv4, ani MIPv6.

4.8 Nepřítomnost kontrolního součtu

Paket protokolu IPv4 nese pole s kontrolním součtem celé hlavičky. Jelikož některá z polí hlavičky by se mohla změnit (např. TTL) na cestě mezi jednotlivými směrovači, musel by se kontrolní součet přepočítávat při každé změně. V dnešních sítích se takové chyby považují za vzácné. Z tohoto důvodu IPv6 nemá žádnou kontrolu. Namísto toho se spoléhá s kontrolou chyb na protokoly spojovací vrstvy. V případě že hlavička je poškozena, nejhorší možný případ je zaslání paketu nesprávnému hostu.

4.9 Adresování

128bitová délka

Hlavní změnou oproti IPv4 je délka síťové adresy. Adresy IPv6 jsou 128 bitů dlouhé, zatímco IPv4 adresy mají 32 bitů. Zatímco IPv4 obsahuje zhruba 4 miliardy adres, IPv6 má dostatek prostoru pro 3.4×10^{38} unikátních adres. Adresy IPv6 se typicky skládají ze dvou logických částí: 64bitová (pod)síťový prefix a 64bitové části hosta, buď automaticky vytvářené na základě MAC adresy rozhraní nebo přiřazené následně. Jelikož globálně unikátní MAC adresa umožňuje vystopovat uživatelské vybavení - a tedy uživatele - IPv6 adresy se mění s časem. RFC 3041 byla vyvinuta s cílem snížit šanci trvalého svázání identity uživatele a IPv6 adresy, což obnovuje některé z možností anonymity existující u IPv4. RFC 3041 popisuje mechanismus na jehož základě náhodné na čase závislé bity mohou být využity k identifikaci okruhu rozhraní, nahrazujíc tak neměnné a sledovatelné MAC adresy.

5. Duality of Service QOS

Pojem kvalita služby (Quality of Service, QoS) vyjadřuje jeden z trendů vývoje technologií a služeb počítačových sítí, poskytovat uživatelům služby s definovanou kvalitou.

V poslední době dochází k rozvoji služeb, jejichž úspěšnost z pohledu uživatele významně závisí na časových charakteristikách komunikace přes počítačovou síť. Jde například o služby Internet telefonie, videokonference a další interaktivní služby. Uživatel požaduje, aby mu byla poskytnuta určitá definovaná kvalita služby (QoS). Například videosignál určitého rozlišení o určitém počtu obrázků za sekundu, zvuk určité šířky pásma nebo přenos souboru dat v určitém čase.

Zmíněné požadavky na kvalitu služeb, které požadují jednotlivé aplikace, lze zaručit pokud se vhodným způsobem aplikuje na počítačovou síť. Nejdůležitější parametry na kterých jsou jednotlivé uživatelské aplikace závislé jsou tyto:

- Šířka pásma (bandwidth)
- Zpoždění (delay)
- Kolísání zpoždění (jitter)
- Ztrátovost (packet-loss)
- Doručení mimo pořadí

QoS počítačové sítě může být implementováno v různých vrstvách v rámci modelu počítačové sítě. Nejčastěji se používá implementace buď na úrovni ATM (je-li technologie ATM použita) nebo na vrstvách v TCP/IP modelu a to síťové a transportní vrstvě. Při implementaci na úrovni vrstev TCP/IP existují dva hlavní přístupy řešení: integrované služby (integrated services, ve zkratce intserv) a rozlišované služby (differentiated services, ve zkratce diffserv). Historicky byla pozornost nejprve věnována QoS na úrovni ATM, později se pozornost přesunula na QoS na úrovni TCP/IP vrstev. Souvisí to s celkovým ústupem od používání technologie ATM.

5.1 Důvody sledování provozu na síti

Důvody sledování provozu na síti mohou být různé. Zde jsou uvedeny některé základní z nich:

- zachování dostupnosti kritických zařízení v sítích,
- zjištění míry využití šířky pásma,
- odstranění problémů se směrováním,
- odhalení bezpečnostních děr.

5.2 Úrovně QoS

Best-effort – jedná se o službu klasického internetu, která poskytuje základní konektivitu bez garancí

Differentiated – služba, která rozděluje prvoz v síti do tříd podle požadavků

CoS – každá třída obsahuje konfigurovatelný QoS mechanismus, podle kterého je potom daná třída obsloužena.

Soft CoS – je bez garancí pro službu jako takovou. Diferencuje pouze provoz.

Guaranteed – požadavek pro přidělení určitého množství síťových zdrojů

Hard QoS – vyžaduje přísné garance od sítě

5.3 QoS na linkové vrstvě

Některé technologie z linkové vrstvy přímo podporují QoS. Především ATM technologie. Bohužel tato metoda se nedostane ke koncovému uživateli, neboť se používá především na páteřní síti. U koncových uživatelů je nejrozšířenější technologií ethernet, který pomocí specifikace 802.1p rozšířil možnost používání QoS v ethernetu. Dalšími možnostmi jsou Frame Relay a Token Ring.

Všechny výše uvedené technologie se uplatňují především v sítích LAN či WAN, pokud na danou problematiku nahlížíme ze strany problematiky QoS. Bohužel se nejedná o řešení, které by postačovalo pro celý Internet.

5.4 QoS z hlediska End-to-End konektivity

End-to-End konektivita začíná na síťové vrstvě. Na daném faktu se podílí velké množství technologií linkové vrstvy, které jsou heterogení. Internet je založený na TCP/IP, proto je pro něj QoS na IP. Funguje tak, že mapuje své funkce QoS do mechanismů linkové vrstvy.

6. QoS ve vrstvách TCP/IP modelu

6.1 Historie

Rozlišení paketu a následné jeho zpracování v rámci QoS je za neseno v RFC 791. Dané informace byly zaneseny v záhlaví IP paketu v poli ToS (Type of Service). V dobách vzniku internetu nebyla podpora ToS důležitá a proto ji většina IP implementací ignorovala.

Na začátku devadesátých let dvacátého století se začaly uplatňovat mechanismy pro plánování front paketů na routrech – WFQ a WRED. Úkolem těchto mechanismů bylo předcházet zahlcení routerů. Dále byl zaveden model integrated services, který splňuje požadavky aplikací, které mají striktní požadavky na šíři pásma. Tento model má garantovat prediktivní chování sítě těmto aplikacím. Jedná se o protokoly RVSP, COPS a další. Zatímco u integrated services signalizují požadavky na specifické QoS aplikace, u následujícího modelu differentiated services síť rozpoznává jednotlivé třídy, které vyžadují specifický QoS. Požadavky na differentiated services jsou zaznamenány ve specifikaci RFC 2430.

6.2 Modely pro IP Qos

- Best-effort
- Integrated services – IntServ
- Differentiated services – DiffServ

6.2.1 Integrated services – IntServ model

IntServ (Integrated Services) je založen na rezervaci pásma. Ta je vytvářena v době shodné s dobou navazování spojení. Danou technologii musí podporovat nejen koncové uzly, mezi kterými se spojení vytváří, ale i všechny prvky třetí vrstvy OSI, uzly sítě jako jsou například směrovače, přes které spojení prochází.

Pokud se nalezne v síti skupina směšovačů, které nepodporují RVSP, tak v daném místě dojde k přerušení rezervace. Signalizace je přes daný směrovač přenesena transparentně a rezervace je vytvořena na následujících uzly, které již RVSP podporují. Pokud mají spoje bez rezervace dostatečnou kapacitu, nemusí dojít k ovlivnění real-time služeb.

Nejznámějším příkladem technologie IntServ je Ressource Reservation Protocol – RSVP. Tento protokol není směrovací, ale kontrolní protokol. Zajišťuje předávání informací mezi sousedními uzly a alokaci pásma pro konkrétní proces.

RSVP zajišťuje pásmo pouze simplexně. V praxi to znamená, že aplikace, která požaduje rezervaci pásma, požaduje tuto rezervaci pouze jedním směrem. Druhý směr si musí zajistit protější uzel. Tato vlastnost je implementována z důvodu, že RSVP je používáno převážně pro multicastové přenosy. Ty se vyznačují asymetrickými požadavky na pásmo. Může dojít pouze k jednosměrné rezervaci pásma.

IntServ je zajímavou metodou pro garantování pásma určitým procesům, ale má několik základních nevýhod:

- nedostatečná škálovatelnost – několik požadavků může zabrat celé dostupné pásmo. To je rezervováno i v době, kdy není procesem zcela využíváno.
- neúplná podpora aplikacemi a operačními systémy – rezervaci pásma si zajišťuje aplikace
- není zajištěna priorizace
- do paketově orientovaného modelu zavádí model okruhově orientovaný
- dosti náročná signalizace mezi sousedními uzly – zvýšení režie

6.2.2 Differentiated Services - DiffServ

Struktura této technologie je založena na jednoduchém modelu, kde datové toky jsou sdruženy do tříd podle stejného typu služby, takže síťové prvky se nemusí starat o každý datový tok zvlášť. Každý IP paket vstupující do počítačové sítě je označen značkou, která udává, jak má být s daným paketem naloženo, nebo-li určuje třídu přenosu poskytnutou paketu.Označení paketů probíhá pouze na vstupu do počítačové sítě. Během přenosu paketů v síti další směšovače pouze přečtou značku každého paketu a dle této značky se řídí plánování paketu.

6.3 Klasifikace paketů

Rozhlehlé počítačové sítě (například národní sítě) bývají obvykle tvořeny propojením sítí menšího rozsahu (například metropolitní sítě). Každá síť může být vybavena jinými směrovači a organizačně řízena jiným subjektem. Z toho vyplývají i různé možnosti zpracování procházejících paketů s ohledem na zajištění požadované QoS. Proto je rozlehlá síť rozdělena na oblasti se samostatnou správou rozlišovaných služeb, na tzv. diffserv domény. Pakety vstupují do diffserv domény přes tzv. ingress směrovače, prochází přes vnitřní směrovače diffserv domény a vystupují přes tzv. egress směrovače. Jsou-li dvě diffserv domény propojeny jedním směrovačem, pracuje egress směrovač jedné diffserv

domény zároveň jako ingress směrovač druhé diffserv domény a plní i opačné funkce pro pakety procházející v opačném směru.

Klasifikace paketů, t. j. označení značkami probíhá na ingress směrovači a to jak při vstupu paketů do sítě, tak i při přechodu z jedné diffserv domény do jiné. Výběr značky při vstupu paketu do sítě může být proveden na základě protokolu, IP adresy odesílatele a adresáta, čísel portů a dalších kriterií, včetně výsledků měření dynamických vlastností přicházejících dat. Při přechodu do jiné diffserv domény může být značka zachována, změněna na jinou značku se stejným významem (pokud různé diffserv domény používají různé značky pro stejné zpracování paketů) nebo změněna na jinou značku s jiným významem (pokud následující diffserv doména nemůže zajistit zacházení s pakety použité v předcházející doméně). Pakety mohou být klasifikovány také již aplikací posílající pakety do sítě. První ingress směrovač může tuto klasifikaci zachovat nebo změnit.

Způsob označení paketu závisí na technologii nebo protokolu použitém pro přenos paketů. Značka může být buď obsažena uvnitř hlavičky paketu, pokud je tam pro ni vhodné místo, nebo připojena vně paketu. Nejčastější je implementace diffserv na úrovni síťové vrstvy modelu sítě při použití protokolu IP. V tomto případě je značka obsažena v poli označeném DS (differentiated services) , které je uloženo buď v místě určeném pro pole TOS (type-of-service) hlavičky protokolu IP verze 4 nebo v místě pro pole Traffic Class hlavičky protokolu IP verze 6. Pole TOS bylo původně určeno pro obdobné účely, související se zpracováním paketu ve směrovačích, nebylo však běžně využíváno. Pole Traffic Class bylo navrženo pro určitý způsob klasifikace paketů bez bližší specifikace. Obě místa jsou tedy vhodná pro uložení značky diffserv. Pole DS má 8 bitů, z toho je 6 bitů určeno pro vlastní značku (differentiated services codepoint, DSCP) a zbývající 2 bity jsou rezervovány pro budoucí použití.

Jiným způsobem označení by mohlo být použití čísel virtuálních spojů v technologii ATM - VCI (virtual channel identifier) a VPI (virtual path identifier). Technologie ATM však používá jinou metodu pro zajištění QoS počítačové sítě.

6.4 Zpracování paketů

Zpracování paketů ve směrovačích v rámci diffserv domény lze popsat ve třech úrovních abstrakce.

Na nejvyšší úrovni je zpracování určeno službou definovanou mezi koncovými body - účastníky komunikace. Služba je definována souborem parametrů popisujících vazbu mezi účastníkem a sítí (Service Level Agreement, SLA). Příkladem služby je tzv. virtuální pronajatý okruh (virtual leased line). Účastník (aplikace) má k dispozici komunikační kanál, který se chová jako skutečný pronajatý okruh, tedy poskytuje stálou předem dohodnutou propustnost s nízkou latencí a malou ztrátovostí paketů, ačkoliv je realizován pomocí prostředků (linek a směrovačů) sdílených spolu s dalšími účastníky.

Na střední úrovni je zpracování paketů určeno tím, jak je s pakety zacházeno jednotlivými směrovači, bez ohledu na ostatní směrovače (per-hop-behaviour, PHB). Každý směrovač totiž zpracovává pakety zcela nezávisle na ostatních směrovačích. Stará se pouze o to, aby jeho vlastní zpracování paketů odpovídalo značkám paketů. Uvnitř diffserv domény nejsou udržovány žádné virtuální spoje přes několik směrovačů. V současné době jsou standardizována dvě PHB - expedited forwarding (EF) a assured forwarding (AF). Diffserv domény mohou implementovat i jiná lokálně definovaná PHB. Specifikace PHB je klíčovou částí diffserv. Předpokládá se, že směrovače budou poskytovat určitá PHB, která budou aplikace využívat k implementaci různých komunikačních služeb (například zmíněný virtuální pronajatý okruh). Hodnota 0 v poli DS je navržena pro označení default PHB, kterým je best-effort přístup.

Specifikace PHB popisuje zpracování paketů z pohledu vnějšího pozorovatele. Určité PHB může být v rámci směrovače implementováno různým způsobem. Tato implementace je nejnižší úrovní popisu zpracování paketů. Možné metody implementace budou uvedeny dále v části věnované řízení front.

6.4.1 Expedited forwarding (EF)

EF PHB zajišťuje, že každý směrovač v diffserv doméně odesílá pakety zařazené do EF PHB průměrnou rychlostí alespoň rovné stanovené rychlosti. Průměrná rychlost se měří v jakémkoliv časovém intervalu delším nebo rovném době potřebné pro odeslání paketu maximální délky stanovenou rychlostí. EF PHB je vhodné pro implementaci virtuálního pronajatého okruhu.

6.4.2 Assured forwarding (AF)

AF PHB umožňuje zařadit pakety do jedné ze čtyř tříd. Každé třídě je ve směrovačích přidělen určitý objem prostředků, například velikost vyrovnávací paměti nebo kapacita výstupní linky. V rámci každé třídy pak může být každému paketu přiřazena jedna ze tří priorit zahození paketu (drop precedence), ke kterému může dojít v případě zahlcení. Směrovač musí odeslat paket mající nižší hodnotu priority se stejnou nebo vyšší pravděpodobností než paket mající vyšší hodnotu priority. AF PHB se používá pro implementaci služeb, u kterých je potřeba volitelná úroveň kvality přenosu.

6.4.3 Traffic conditioning

Při provozu počítačové sítě se běžně stává, že objem dat přicházejících na určitý směrovač, která mají být dále poslaná na určitý výstupní port směrovače, přesahuje okamžitou kapacitu linky připojené na daný výstupní port. Přicházející pakety jsou v tom případě ukládány do fronty na směrovači. Je však třeba vyřešit situaci, kdy dojde k vyčerpání kapacity fronty (congestion management) nebo předejít vyčerpání kapacity fronty (congestion prevention). Dále je potřeba upravovat objem dat přicházejících do diffserv domény na ingress směrovači v souladu s dohodnutou SLA a implementovat požadovaná PHB na všech směrovačích. Úprava objemu přicházejících dat se obvykle provádí jen na ingress směrovači. K tomu účelu se používají techniky nazvané policing a traffic shaping.

Značkování paketů se provádí na základě klasifikace a může být ovlivněno i měřením dynamických vlastností přicházejících dat, stejně jako policing a traffic shaping. Zpracování

paketů následující po klasifikaci se souhrnně nazývá traffic conditioning a soubor parametrů popisujích toto zpracování se nazývá Traffic Conditioning Agreement (TCA).

Policing obvykle představuje jednoduché zahazování přicházejících paketů (paket je směrovačem ignorován, není poslán dále a o zahození není poslána žádná zpráva), které se začne provádět při dosažení určité podmínky. Tou může být vyčerpání kapacity fronty nebo překročení určitého objemu přicházejících dat.

Inteligentnější metodou je traffic shaping, při které jsou směrovačem jednotlivé pakety pozdržovány tak, že se změní průběh okamžitého objemu odesílaných dat v čase ve srovnání s přijímanými daty, z čehož vyplývá název této techniky. Obvykle se průběh vyrovná tak, že data přicházející na směrovač nepravidelnou rychlostí (s krátkými špičkami - bursts) jsou odesílána stálou rovnoměrnou rychlostí odpovídající části kapacity výstupní linky.

K tomuto vyrovnávání se používá metoda, které se říká "token bucket". Metoda logicky patří do komponentu měření a může být použita i pro ovlivnění značkování nebo rozhodnutí o policing (zahození paketu). Token bucket je nádoba obsahující v každém okamžiku určitý počet tokenů, každý z nich je povolením k odeslání nebo jinému zpracování určitého objemu dat, například 1 bajt. Na počátku je nádoba plná. Při příchodu každého paketu je ověřeno, zda počet tokenů v nádobě alespoň odpovídá velikosti paketu. Pokud ano, paket je zařazen do fronty k odeslání na výstupní port nebo určitým způsobem označen. Zároveň je z nádoby odebrán počet tokenů odpovídající velikosti paketu. Pokud ne, paket je zahozen nebo označen jiným způsobem. Tokeny jsou do nádoby plynule doplňovány stálou rychlostí, dokud není nádoba plná. Token bucket lze tedy popsat dvěma parametry: rychlostí doplňování tokenů *r* (obvykle dle kapacity výstupní linky) a velikostí nádoby *b*. Je zřejmé, že dlouhodobý průměr rychlosti přicházejících dat nesmí být vyšší než je rychlost doplňování tokenů a krátkodobé špičky nesmí překročit velikost nádoby (může jít o jednu velkou špičku nebo více menších krátce po sobě následujících špiček), jinak dojde k zahození paketů nebo jiné odpovídající akci. Je také možné použít více nádob s různými parametry paralelně a tak třídit pakety do více kategorií. To může být použito například pro nastavení drop precedence u AF PHB.

6.5 Řízení front

Při správné funkci traffic conditioning na ingress směrovači by na vnitřních směrovačích již nemělo dojít k vyčerpání kapacity front a proto tyto směrovače již pouze implementují požadovaná PHB. Implementace PHB však musí být provedena i na ingress směrovači, kde následuje po traffic conditioning. Jednotlivá PHB jsou typicky realizována použitím více front s určitým algoritmem pro zařazování přicházejících paketů do jednotlivých front a pro obsluhu front, tedy výběr paketů z front pro odeslání.
Nejrozšířenější metody pro správu více front jsou priority queueing (PQ), class-based queueing (CBQ) a weighted fair queueing (WFQ). Ve všech třech případech je každý přicházející paket vložen do jedné z několika front podle své značky. Metody se liší v tom, jak jsou jednotlivé fronty obsluhovány.

6.5.1 Priority queueing (PQ)

U PQ má každá fronta přiřazenu určitou prioritu, která je u každé fronty jiná. Platí, že z fronty s určitou prioritou může být vybrán paket k odeslání jen pokud jsou všechny fronty s vyšší prioritou prázdné. Fronty s vyšší prioritou mají tedy absolutní přednost před frontami s nižší prioritou. Této vlastnosti lze využít například pro implementaci expedited forwarding PHB (EF PHB). V tomto případě stačí dvě fronty - jedna pro EF PHB, druhá pro "best-effort" provoz. Nevýhodou PQ je možnost uváznutí (starvation) dat ve frontě s nižší prioritou. Je-li zpoždění paketů příliš veliké, jsou navíc v protokolu vyšší vrstvy obvykle považovány již za ztracené a mohou být poslány znovu, čímž dále zatíží počítačovou síť.

6.5.2 Class-based queueing

U CBQ jsou jednotlivé fronty obsluhovány cyklicky - jedna po druhé (round-robin). Jakmile určitá fronta přijde na řadu, jsou z ní odesílány pakety dokud není odeslán alespoň stanovený minimální počet bajtů nebo dokud není fronta vyprázdněna. Počet bajtů odeslaných při jedné obsluze fronty lze stanovit individálně pro každou frontu. CBQ může být použito například pro implementaci assured forwarding PHB (AF PHB). V tomto případě jsou potřeba čtyři fronty - jedna pro každou třídu AF PHB. CBQ však musí být ještě doplněno dalším mechanizmem pro implementaci drop precedence jednotlivých tříd AF PHB, například WRED.

6.5.3 Weighted fair queueing (WFQ)

U WFQ jsou průběžně obsluhovány vždy všechny fronty. Žádná fronta nemá vyšší prioritu než jiná fronta. Jednotlivým frontám je přitom přidělována alikvotní část kapacity výstupní linky podle váhy přiřazené ke každé frontě. Není-li však kapacita přidělená určité frontě právě využívána, může být využita pro obsluhu jiné fronty. Přesný postup při určení, ze které fronty má být vybrán další paket k odeslání, je následující. Pro každý přicházející paket je vždy vypočten čas, kdy nejpozději musí být daný paket odeslán vzhledem k tomu, že pro obsluhu fronty, do které je paket zařazen, je garantována určitá kapacita linky.

Například, je-li paket vkládán do fronty obsluhované rychlostí 1 paket za 2 jednotky času (pro jednoduchost uvažujeme pakety stejné délky), přičemž ve frontě již jsou uloženy 2 pakety, potom tento nový paket musí být obsloužen nejpozději za 6 jednotek času od jeho příchodu. Po ukončení obsluhy každého paketu se vždy určí, který paket ze všech paketů na vrcholu neprázdných front má být obsloužen nejdříve. Tento paket je pak vybrán pro odeslání. WFQ může být spolu s WRED, obdobně jako CBQ, použito pro implementaci AF PHB.

6.5.4 (Weighted-) Random Early Detection (RED / WRED)

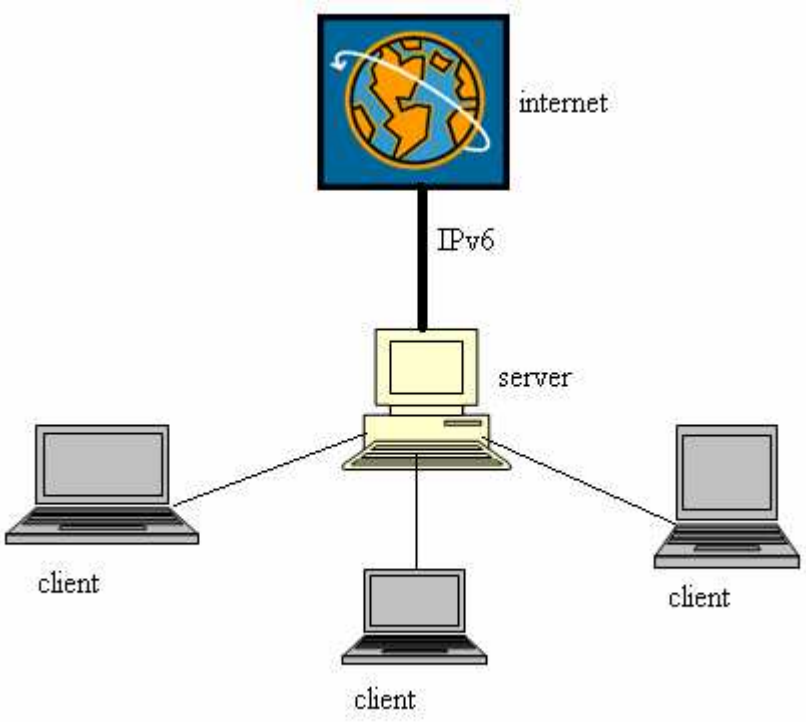
RED nebo WRED jsou metody prevence zahlcení (congestion prevention). Pokud necháme fronty zcela naplnit pakety a teprve potom začneme zahazovat další příchozí pakety, může dojít k synchronizaci zahlcení mezi více směrovači a vytvoření vln střídajícího se zahlcení s okamžiky, kdy je snížen objem dat posílaných do sítě tak, že není využita kapacita linek. Příčinou tohoto jevu je chování protokolu TCP, kterým se dnes přenáší významná část všech dat v síti Internet. Pokud totiž protokol TCP na straně odesílatele detekuje ztrátu paketu, odešle ztracený paket znovu a zároveň sníží rychlost odesílání paketů k příjemci neboť předpokládá přetížení přenosové cesty a snížením rychlosti chce předejít dalším ztrátám paketů. Protože směrovačem obvykle prochází řada různých TCP spojení současně od různých odesílatelů, dojde při zahlcení směrovače ke snížení rychlosti na řadě odesílatelů a tím i k následnému nevyužití kapacity výstupní linky směrovače. Jednotliví odesílatelé postupně obnoví původní rychlost přenos a dojde znovu k zahlcení směrovače.

Tomuto jevu se snaží předejít metoda RED. Přesáhne-li naplnění fronty určitou mez, začne směrovač zahazovat pakety z náhodně vybraných TCP spojení. Pravděpodobnost zahození paketu se zvyšuje se zvyšujícím se naplněním fronty. Tím dojde ke snížení objemu dat od některých odesílatelů a plynulému vyrovnání celkového objemu přicházejících dat s kapacitou odchozí linky. U modifikované metody WRED závisí pravděpodobnost zahození paketu též na značce paketu přidělené klasifikací. Limit naplnění fronty při jehož překročení může být paket zahozen je různý pro pakety s různými značkami.

[3]

7. Praktická část

V praktické části bude navržena síť, která bude fungovat pod IP verzí 6. Návrh bude vyplývat z klasické hvězdicovité topologie, kdy všichni klienti jsou připojeni k jednomu prvku – serveru, který je poté součástí globální sítě internet.
Na server budou připojeni nejméně dva klienti. Jeden bude virtuální a druhý reálný. Síť bude navržena a provozována pod distribucí Arch Linux. Jako server bude sloužit jakýkoliv počítač, který bude nakonfigurovaný jako server. Celá síť bude implementována pod IP verzi 6. Na serveru bude implementovaná i jedna virtuální stanice klienta. Případné ostatní stanice mohou být také virtuální nebo reálné.



Obr 1: Návrh sítě hvězdicové topologie.

7.1 Konfigurace serveru

7.1.1. Instalace distribuce Linux Arch pro server i klienta

Jako první věc, která nás čeká je instalace distribuce Linux Arch na počítač, který jsme zvolili, že bude sloužit jako server.

Po prostudování požadavků této distribuce na vybavení našeho počítače, které jistě splňuje, neboť procesor i686 nebo x86-64 a 96 MB RAM, už v dnešní době splňují snad všechny fungující počítače, pokud nebereme v úvahu historické kousky z Technického muzea, můžeme stáhnout .iso z ftp serveru, který slouží pro Českou republiku a je umístěn na pražském ČVUT: ftp://ftp.sh.cvut.cz/MIRRORS/arch

Po stažení je možné pomocí příkazu: md5sum --check arch-0.8md5sum možno ověřit integritu staženého .iso obrazu. Po stažení a ověření integrity je zapotřebí daný obraz vypálit na médium CD-R.

Vypálený .iso obraz na mediu vložíme do počítače. Ten restartujeme a při náběhu se dostaneme do BIOS, kde překontrolujeme, zda máme nastavené bootování z média. Pokud nemáme, nastavíme a opustíme BIOS s uložením změn. Dojde k nabootování z CD a objeví se příkazový řádek. Bez jakýchkoliv parametrů a příkazů potvrdíme klávesou ENTER a instalace pokračuje. Načteme mapu kláves. Do příkazové řádky zadáme příkaz

km

a dle potřeby si vybereme klávesnici, která nám vyhovuje a popřípadě font.

Spustíme /arch/setup. Po zobrazení úvodní zprávy budeme dotázáni na typ instalace. Vybereme FTP instalaci, neboť máme relativně rychlé připojení k internetu a vzhledem k tomu, že se budeme zabývat IP verzí 6, tak potřebujeme mít nainstalované nejnovější balíčky a utility. Po vybrání z jedné alternativ další instalace pokračuje obdobně jako při instalaci Windows, kdy veškeré nabídky letmo omrkneme a potvrdíme tlačítkem DONE. Klávesové šipky umožňují i případný návrat k předchozí nabídce.

Jelikož jsem zvolili FTP instalaci musíme v této fázi instalace zvolit, které síťové zařízení použijeme. Pokud není žádné zařízení detekováno je zapotřebí vybrat manuálně či jej případně načíst. Po načtení aktivní síťové karty zvolíme YES, pokud používáme DHCP server. Funkčnost se dá ověřit příkazem ping.

Dalším krokem je příprava pevného disku. Ponecháme volbu Auto-Prepare. Ta má za úkol rozdělit disk na boot, swap a root oddíl, na kterých vytvoří souborové systémy. Tyto oddíly budou v systému připojeny na správné místo. Přesně bylo vytvořeno:

32 MB ext2 /boot oddíl
256 MB swap oddíl
Root a /home oddíly ze zbývajícího místa

Při pokračování v instalaci se dostáváme k výběru balíčků. Položka Select Packages nám umožňuje v našem případě vybrat balíčky z FTP, které chceme instalovat. Při dotazu, ze kterého FTP zrcadla chceme instalovat jsme vybrali první nabízenou možnost. Došlo k načtení balíčků, které jsou zobrazeny a rozděleny do různých kategorií. Vybrali jsme základní BASE, neboť počítač bude neustále připojen na internet a není problém kdykoliv stáhnout a doinstalovat potřebné balíčky. Po výběru došlo k instalaci balíčků. Pokračujeme automatickou detekcí hardware. Pro pozdější editaci jsme vybrali editor Nano. Krok úpravy jednotlivých souborů jsme přeskočili.
Následuje instalace jádra v našem případě se jedná o jádro 2.6.33. Při výběru zavaděče jsme zvolili LILO, který se nainstaloval automaticky. Další příkaz je Exit Install. Po jeho zvolení jsme vyjmuli CD z mechaniky. V příkazové řádce jsme napsali reboot.
Po nabootování systému jsme se přihlásili jako root bez hesla. Ihned po přihlášení jsme nastavili heslo příkazem

passwd.

Následovalo nastavení našeho internetového připojení a ověření jeho funkčnosti.

Velikou výhodou zvoleného jádra je, že veškeré požadavky na IP verzi 6 a potřebné utility jsou v tomto jádře implementovány a defaultně zapnuty, tudíž není potřeba kompilovat jádro.

7.1.2 Získání adresy IPv6

Pro náš návrh sítě je důležité získat pro server adresu IPv6. Jsou různé možnosti, jak lze danou vytouženou adresu vyzískat. Přeci jen implementace IPv6 je teprve, dalo by se říci v počátcích a ne každý tuzemský operátor umožňuje či poskytuje danou službu. IPv6 rozeznává několik druhů adres, z nichž některé už máme, některé si můžeme vymyslet a jen některé musíme získat oficiální cestou.

Adresy se dělí podle dosahu:

- Linkové
- Místní
- Lokální

Linkové (link-local scope)

Platí jen na konkrétním drátě, třeba na ethernetovém segmentu, na PPP lince, nebo v jednom virtuálním tunelu. Adresa začíná prefixem **fe80::/10** a třeba v případě ethernetu se odvozuje z HW adresy síťové karty. Je to tedy adresa, kterou už máme.

Místní (site-local scope)

Platí v rámci určité sítě, např. v rámci jedné organizace, školy, atd. Adresy tohoto rozsahu začínají prefixem **fec0::/10** a je jen na nás, zda a jak je použijeme. Jde tedy o adresy, které si můžeme vymyslet.

Globální (global scope)

Místní adresy jsou prima, ale nehodí se pro situace, kdy chceme komunikovat i vně své sítě. Např. veřejnému webserveru je celkem moudré dopřát i adresu globální. Ta se oficiálně jmenuje *Aggregatable Global Unicast Address* a začíná prefixem **2000::/3**. Adresu tohoto rozsahu si samozřejmě vymyslet nemůžeme.

Možnosti získání IPv6 adresy:

- 6bone - tunel
- Tunel broker
- TSP = Tunel Setup Protocol

6bone

IPv6 provoz může být buď nativní, což je případ, kdy IPv6 pakety běhají po kabelu přímo, nebo tunelovaný, kdy jsou na jednom konci tunelu IPv6 pakety zabaleny do IPv4 a odeslány na druhý konec tunelu jako standardní IPv4 provoz. Tam jsou opět vybaleny a zpracovány jako kdyby nikdy žádným tunelem nešly.

Tunel můžeme získat způsobem poněkud ponižujícím. Najít si v seznamu sítí jednu, která je připojena přes 6bone. Poté zašleme jejímu správci prosebný mail, zda u něj není možno dostat nějaký volný prefix, který by nám přidělil. Pokud se tak stane, je to fajn a už nám nebrání nic v tom, abychom vybudovali tunel do světa IPv6.

Tunel broker

Tunel broker je služba, která slouží k získání adresy IPv6. Na požádání nám tunel broker alokuje část svého adresního prostoru. Samozřejmě se to neobejde bez vyplnění webového formuláře a registrací. V České republice je vhodné využít tunel brokera XS26, který má i přístupové body v České republice. Proč právě XS26? Je více než vhodné, aby tunel broker byl dosahem co nejblíže naší sítě IPv4, neboť se jedná o další hop z hlediska routování v IPv6.

Problémem může být dynamická adresa, neboť je pak zapotřebí neustále předělávat konfiguraci tunelu.

TSP

Jedná se o protokol, který je určen k nastavování tunelů. Umožňuje rekonfiguraci tunelu i při měnící se dynamické adrese IPv4. Jedním z tunel brokerů, který používá TSP je Freenet6. V praxi po registraci u tohoto tunel brokera obdržíte software, který spustí při každém přihlášení k internetu a on vám překonfiguruje tunel.

V mé diplomové práci nevyužijeme ani jednu možnost tvorby tunelu, neboť síť VUT v Brně podporuje protokol IPv6 díky poskytovateli CESNET. Tudíž budeme na IPv6 připojeni přímo bez pomocí tunelu.

7.1.3 Konfigurace serveru

Momentálně máme nainstalovaný Arch Linux na počítači. Nicméně bez dalších úprav nám tento počítač neposlouží jako server. Pro začátek bylo zapotřebí nainstalovat pár utilit pro běžnou práci jako je například mc či http.

Následně jsme se ujistili, že dané jádro opravdu podporuje protokol IPv6. To můžeme provést několika způsoby. V našem případě jsem danou skutečnost zkontrolovali v grafickém rozhraní pod příkazem

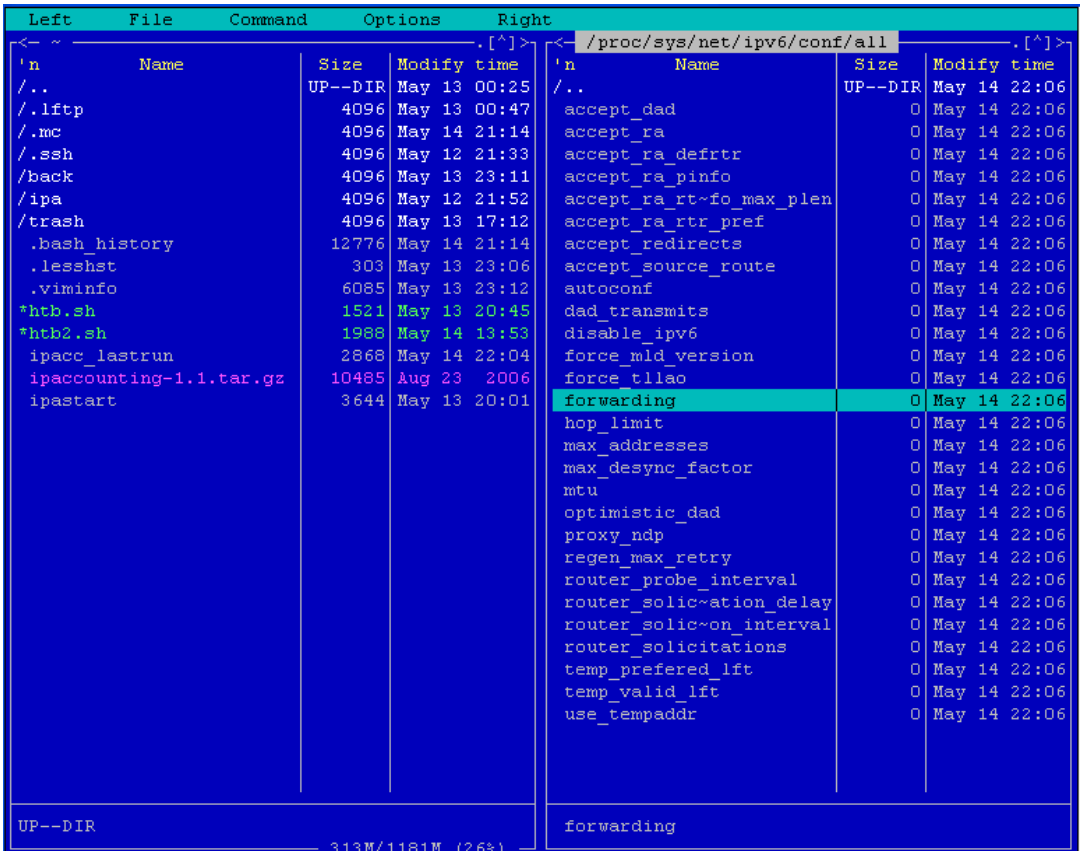
```
# make menuconfig.
```

Ve složce Networking. Zde jsme překontrolovali, zda je opravdu spuštěna podpora IPv6. Dále nás zajímalo, zda je povoleno HTB, CBQ, PRIO, TFB a další pro QoS.

Dalším krokem je nastavení iptables. V našem případě jsme ponechali veškerý provoz povolený. To stejné jsme udělali i v případě iptables6.

Pro komunikaci síťové karty serveru v obou směrech jsme zapnuli forwarding. Zapnutí forwaringu jsme provedli ručně:

V souboru /proc/sys/net/ipv6/conf/all/forwaring jsme nastavili hodnotu na 1.



Obr.2: Povolení forwardingu

Pokud nemáme distribuci Fedora nebo SUSE, kde je ještě zapotřebí povolit forwarding v:

/etc/sysconfig/sysctl

IP_FORWARD = „yes“

IPV6_FORWARD = „yes“

Defaultně jsou tyto hodnoty nastaveny na „no“.

Další potřebná nastavení:

/etc/sysct1.conf

Net.inet6.ip6.forwarding=1

/etc/sysconfig/SUSEFirewall2

FW_ROUTE=“yes“

Pokud jsme zvládli předchozí konfiguraci a nastavení, můžeme se přesvědčit, zda náš počítač získal adresu IPv6.

Přesvědčíme se příkazem:

ifconfig

Ten nám vypíše veškeré ethernetové rozhraní a jejich konfiguraci.

```
[root@myhost ~]# ifconfig
eth0      Link encap:Ethernet  HWaddr 00:50:56:11:0A:14
          inet addr:147.229.150.50  Bcast:147.229.150.255  Mask:255.255.255.0
          inet6 addr: 2001:718:802:9091:250:56ff:fe11:a14/64 Scope:Global
          inet6 addr: 2001:718:802:9096:250:56ff:fe11:a14/64 Scope:Global
          inet6 addr: fe80::250:56ff:fe11:a14/64 Scope:Link
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:719519 errors:0 dropped:0 overruns:0 frame:0
          TX packets:523137 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:1000
          RX bytes:69452526 (66.2 Mb)  TX bytes:575641022 (548.9 Mb)
          Interrupt:18 Base address:0x1080

lo        Link encap:Local Loopback
          inet addr:127.0.0.1  Mask:255.0.0.0
          inet6 addr: ::1/128 Scope:Host
          UP LOOPBACK RUNNING  MTU:16436  Metric:1
          RX packets:763 errors:0 dropped:0 overruns:0 frame:0
          TX packets:763 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:0
          RX bytes:74168 (72.4 Kb)  TX bytes:74168 (72.4 Kb)

[root@myhost ~]#
```

Obr.3: Příkaz ifconfig

Pro naše další nastavení jsou pro nás důležité hodnoty u rozhraní eth0. Co se týče hodnot pro IPv6, tak jsme obdrželi dvě globální adresy a jednu linkovou. Globální IPv6 adresu 2001:718:802:9091:250:56ff:fe11:a14 budeme na klientech používat jako výchozí bránu. Dále nám také poslouží jako adresa ftp6 serveru pro lokální síť.
Důležité je, aby měl počítač přidělenu minimálně jednu globální IP verze 6 adresu, aby mohl komunikovat s okolním IPv6 světem. Adresy máme přiřazeny, tak se tedy dáme do testování průchodnosti do sítě IPv6. Jednou z mála webových adres fungující na IPv6 je www.ipv6.org. Dostupnost ověříme příkazem ping.

```
# ping6  www.ipv6.org
```

```
[root@myhost ~]# ping6 www.ipv6.org
PING www.ipv6.org(igloo.stacken.kth.se) 56 data bytes
64 bytes from igloo.stacken.kth.se: icmp_seq=1 ttl=51 time=44.1 ms
64 bytes from igloo.stacken.kth.se: icmp_seq=2 ttl=51 time=43.2 ms
64 bytes from igloo.stacken.kth.se: icmp_seq=3 ttl=51 time=42.0 ms
64 bytes from igloo.stacken.kth.se: icmp_seq=4 ttl=51 time=42.0 ms
64 bytes from igloo.stacken.kth.se: icmp_seq=5 ttl=51 time=41.8 ms
64 bytes from igloo.stacken.kth.se: icmp_seq=6 ttl=51 time=42.0 ms
^C
--- www.ipv6.org ping statistics ---
6 packets transmitted, 6 received, 0% packet loss, time 5011ms
rtt min/avg/max/mdev = 41.806/42.557/44.180/0.894 ms
```

Obr.4: Příkaz ping6

Nyní máme ověřeno, že naše spojení se světem IPv6 je funkční.

7.1.3.1 FTP6

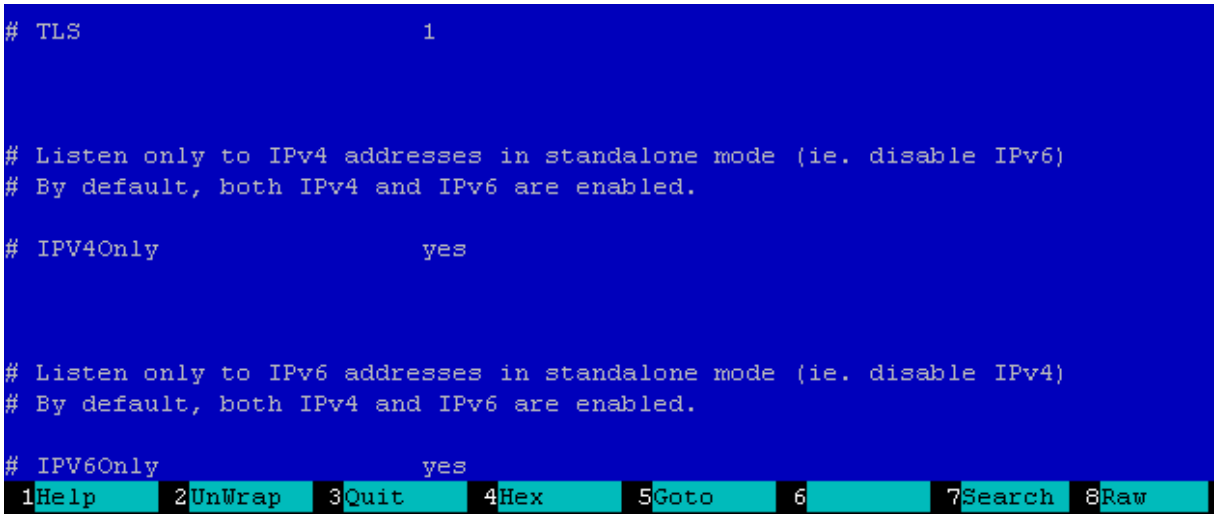
Jednou ze základních služeb každého serveru je FTP server. FTP je nejstarší protokol sloužící pro přenos souborů mezi počítači. Na těch mohou být instalovány různé operační systémy. Využívá porty TCP/20 a TCP/21. Z toho port 20 slouží k přenosu dat.
V našem případě jsme na serveru zprovoznili FTP server, který podporuje IPv6. Instalace proběhla příkazem:

```
# pacman –S pure-ftpd-1.0.29-2
```

Výhodou tohoto ftp serveru je, že umožňuje pracovat pod IPv6 i pod Ipv4. Lze na něm pracovat na obou protokolech bez nějakých velkých úprav v konfiguračním souboru. Pokud ovšem chceme zakázat jeden z protokolů, tak je to možné pomocí nastavení v:

```
# /etc/pure-ftp.conf
```

Na samém konci konfiguračního souboru nalezneme možnost protokol Ipv4 nebo v6 zakázat. V našem případě jsme ponechali oba protokoly povoleny. Viz. obrázek.



Obr.5: Pure-ftp.conf

Uvedení ftp6 serveru do provozu:

```
# /usr/sbin/pure-ftpd -S 21 -B -l unix -l -A -x -j -R -Z
```

-S 21 – server poběží na portu 21

-B – přepnutí serveru do daemon režimu

-1 unix – možnost přihlašování Unixových uživatelů

Nakonec řekneme serveru, že každého uživatele má chrootovat do jeho domácího adresáře, zajistíme přístup ke skrytým souborům, vytvoření domácích adresářů, zakážeme CHMOD anonymním uživatelům a zapneme ochranu proti špatně nastaveným právům.

Výhodou chrootingu je, že případný útočník, který napadne náš počítač přes software, který je v chroot umístěn, nemůže ohrozit celý systém, neboť má přístup pouze k souborům, které potřebuje daný software pro svoji činnost. Jedná se o vytvoření uzavřeného prostředí.

Zbývá už jen nastavit přístupové jméno a heslo:

```
# pure-pw useradd ftp6 -u cx -g cx -d /home/ftp6
```

Na ftp6 jsme umístili soubor, který mohou klienti po přihlášení stahovat.

```
# /home/ftp6/soubor50Mib.dat
```

7.1.3.2 Ipaccounting a rddht

Pro zobrazování grafů a počítání ip provozu bylo zapotřebí zprovoznit ipaccounting a rrdht. Pro Ipaccounting jsme použili verzi 1.1. Její skript jsme poupravili pro protokol IP verze 6. Vesměs šlo jen o drobné úpravy. Typu přepsání iptables na ip6tables. Obdobně tak i skript rrdht.

```
active_iface="eth0"
dir_data_txt="/var/www/html/ipaccounting"
dir_data_rdd="/var/www/html/ipaccounting/rrd"
rdd_active="on"
txt_active="on"
testing_active="on"

# Testovani pravidel a chainu v IP6TABLES
if [ $testing_active == "on" ]; then
    if [ "`ip6tables -t mangle -L POSTROUTING | grep "DATA_IN"`" == "" ]; then
        echo "Pridavam chain DATA_IN"
        ip6tables -t mangle -N DATA_IN
        ip6tables -t mangle -A POSTROUTING -j DATA_IN
    fi
    if [ "`ip6tables -t mangle -L PREROUTING | grep "DATA_OUT"`" == "" ]; then
        echo "Pridavam chain DATA_OUT"
        ip6tables -t mangle -N DATA_OUT
        ip6tables -t mangle -A PREROUTING -j DATA_OUT
    fi
```

Obr.6: Ipaccounting

7.1.3.3 HTB

Při sestavování skriptu HTB jsme vycházeli z již vytvořených skriptů pro IP verzi 4. Pouze pravidla pro iptables jsme museli předělat pro verzi ip6tables.

Disciplína vzniklá na základě CBQ (Clase Based Queueing). Pro jednotlivé třídy je možnost nastavit parametry rate a ceil. Ceil je celkové maximum, rate je garantované minimum pro danou třídu. HTB neumožňuje odesílat pakety dané třídy rychleji, než je dáno hodnotou ceil. Takže je možné vytvořit třídu, která bude pracovat na rychlosti 256kbit/s. Server pak danou třídu bude obsluhovat maximálně touto zadanou rychlostí.

Kořenovou rychlost si pak klienti dělí mezi sebou dle pravidel popsaných ve skriptu. Skript je přiložen v příloze.

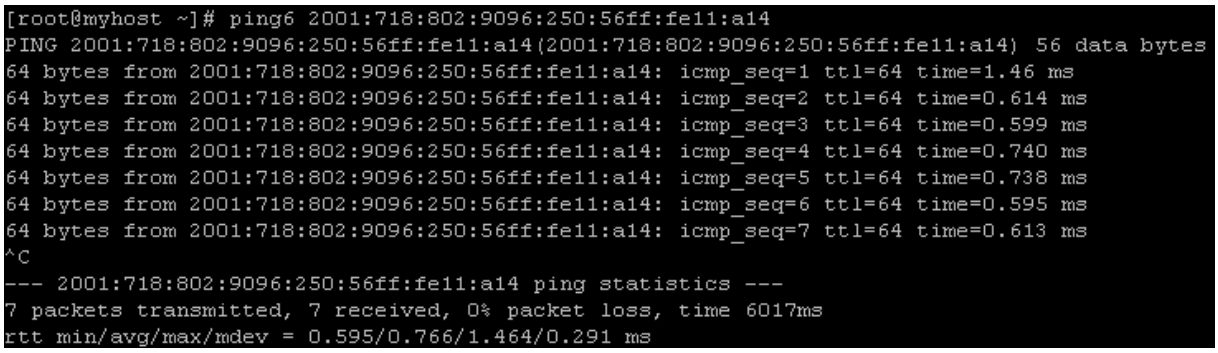
7.1.4 Konfigurace klienta

Klient nemusí mít instalován Arch Linux jako server v našem případě, ale může být na něm nainstalovaná jakákoliv jiná distribuce Linux. Na síti, kterou jsme měli ke svému experimentu k dispozici jsme měli jednoho klienta s Arch Linuxem, jednoho se SUSE a zbývající klienti pracovali na distribuci Fedora.

Na počítači klienta je zapotřebí ověřit, zda dané jádro podporuje IPv6. To jsme ověřili obdobně jako u serveru. Záleží na jednotlivých distribucích jaké příkazy jsou použity. Po případném zapnutí podpory v jádře a jeho kompilaci ověříme příkazem

```
# ping6 2001:718:802:9096:250:56ff:fe11:a14
```

zda klient komunikuje s naším serverem.



Obr.7: Ping6 od klienta na server

Předešlý obrázek nám potvrdil, že náš klient komunikuje s naším serverem. Nyní ještě potřebujeme nastavit, aby klient používal jako výchozí počítač náš server. V příkazu route mu nastavíme náš server jako výchozí bránu.

Routování klienta přes náš server vyřešíme příkazem:

```
# route –A inet6 add default gw 2001:718:802:9096:250:56ff:fe11:a14
```

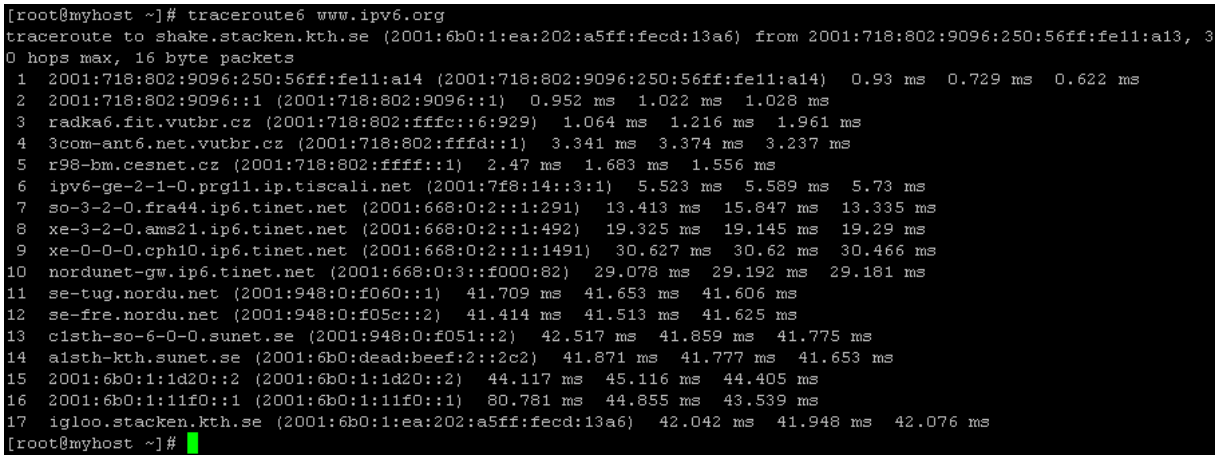
Daným příkazem jsme docílili toho, že veškerý provoz v IPv6 půjde přes server. Nyní překontrolujeme, zda klient dostal přidělenou globální adresu a linkovou adresu IP verze 6. K tomu požijeme příkaz

```
# ifconfig
```

obdobně jako při zjišťování na serveru.

Po kontrole obdržení adres IPv6 můžeme ověřit, zda jsme opravdu docílili kýženého přesměrování klienta přes server, nebo-li zda námi nakonfigurovaný server pracuje jako výchozí brána pro námi nakonfigurovaného klienta/klienty.
Ověříme příkazem:

```
# traceroute6 www.ipv6.org
```



Obr.8: Příkaz traceroute6

Jak je vidět na předchozím obrázku, první hop jde přes počítač, který má adresu IPv6 2001:718:802:9096:250:56ff:fe11:a14, tzn. veškerá komunikace daného klienta prochází přes ethernetové rozhraní našeho serveru.

Takto nakonfigurovat můžeme nezměrné množství klientů, pokud to dovolí nastavení našeho serveru.

7.1.4.1 Připojení klienta na ftp6

Pokud máme správně na serveru nakonfigurovaný v našem případě ftp6 server, máme na něm zřízen přístup pro uživatele a jsou zadána práva uživateli k jeho užívání, nebrání už nic klientovi aby se na daný server přihlásil.

To provedeme příkazem:

```
# lftp –u ftp6 2001:718:802:9091:250:56ff:fe11:a14
```

Vyskočí požadavek na zadání hesla. Po jeho vyplnění je klient přihlášen na ftp6 server. Zde se může pohybovat jako v každé Linuxové struktuře například příkazy ls, cd a podobně.

```
[root@myhost ~]# lftp -u ftp6 ftp://2001:718:802:9096:250:56ff:fe11:a14
Password:
lftp ftp6@2001:718:802:9096:250:56ff:fe11:a14:~> ls
drwx------    2 ftp6       users            1024 May 13 01:27 .
drwx------    2 ftp6       users            1024 May 13 01:27 ..
-rw-------    1 ftp6       users             155 May 13 01:27 .bash_history
-rw-r--r--    1 ftp6       users              16 Aug  1  2009 .bash_profile
-rw-r--r--    1 ftp6       users             108 Aug  1  2009 .bashrc
-rw-r--r--    1 0          0                  28 May 13 00:45 ahoj.txt
-rw-r--r--    1 0          0           50000000 May 13 01:27 soubor50Mib.dat
lftp ftp6@2001:718:802:9096:250:56ff:fe11:a14:/>
```

Obr.9: FTP6 – přihlášený klient

Nyní může klient stáhnout libovolný soubor, který mu to dle nastavení práv umožní. V našem případě budeme stahovat soubor s názvem soubor50Mib.dat.

Spustíme příkazem:

```
# get soubor50Mib.dat
```

Následně se spustí stahování daného souboru. Na obrazovce vidíme jakou rychlostí je soubor stahován a kolik procent je již staženo. Po stažení souboru zůstane klient připojen k ftp6 serveru. Pro odhlášení je nutno zadat příkaz

```
# quit.
```

```
lftp ftp6@2001:718:802:9096:250:56ff:fe11:a14:/> get soubor50Mib.dat
soubor50Mib.dat' at 3240132 (6%) 24.1K/s eta:32m [Receiving data]
```

Obr:10: Stahování souboru z ftp6

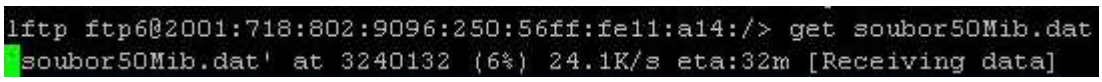
Po zadání výše uvedeného souboru se automaticky dostaneme do adresářové struktury Linux v Midnight comanderu do adresáře /root, kde máme stažený soubor uložen.

7.2 Testování QoS

Momentálně máme nakonfigurovaný server, klienty, spuštěný ftp6 server, zajištěný ipaccounting, spuštěný skript HTB pro omezování provozu, který je aplikován na serveru a zajištěno generování grafů přes RRD.
Nezbývá nám tedy nic jiného, než ověřit, že jsme vše nastavili správně a vše funguje, jak by mělo.

7.2.1 Jeden klient

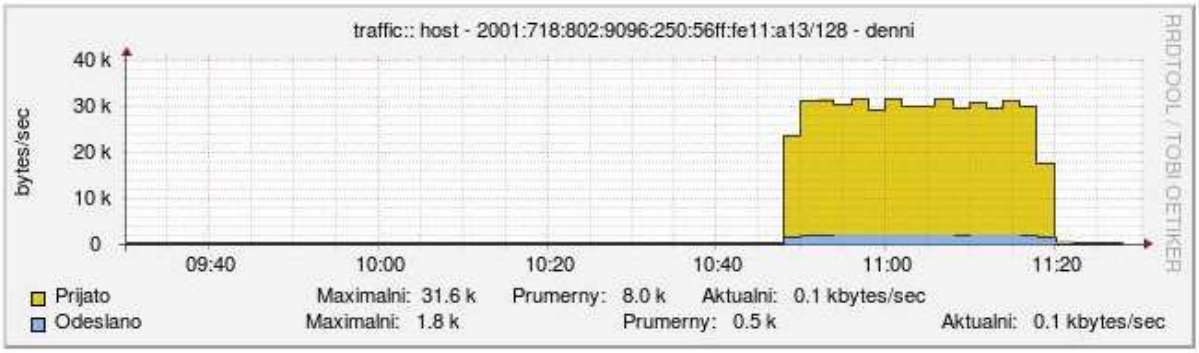
Prvotní ověření nastavení jsme se rozhodli provést na stanici virtuálního klienta na distribuci Arch Linux. Stejnou distribuci máme i na serveru. Po nastavení route, ověření konektivity a kontrole, zda opravdu veškerá komunikace jde přes námi definovaný server jsme přihlásili klienta na ftp6 server a zahájili stahování souboru soubor50Mib.



Obr.11: Stahování souboru jedním klientem

Jak můžeme vidět na obrázku, tak při stahování jedním klientem dosahuje klient momentální rychlosti 24.1kbyte/s což je po přepočtu 192,8 kbit/s. Pokud tedy máme v HTB nastavený ceil 256 kbit/s, měl by teoreticky klient pracovat na této hodnotě.

Nicméně nic není dokonalé a i stroje mají svoji odchylku. Nehledě na to, že na daném klientovi běžel další ethernetový provoz, který danou rychlost také mohl ovlivnit a ovlivnil. A v námi vypočtené hodnotě není započtena hodnota odchozího provozu.



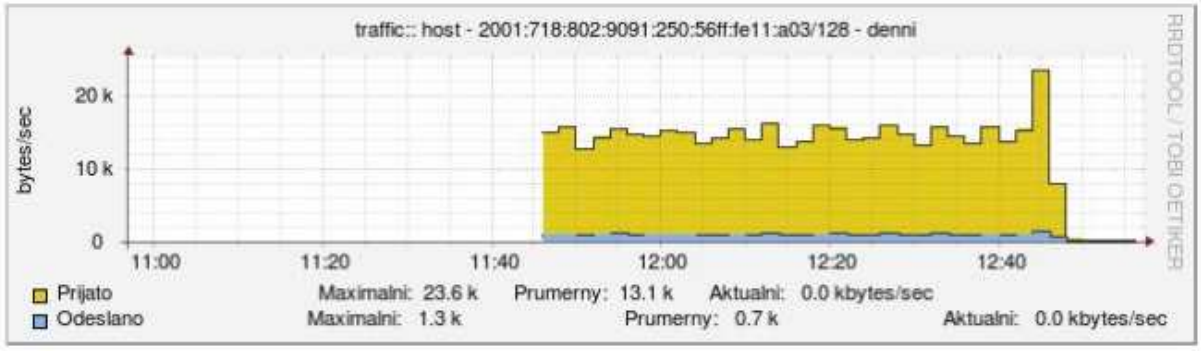
Obr.12: Graf stažení souboru jedním klientem

Na výše vyobrazeném grafu je znázorněn celý průběh stahování daného souboru jedním klientem. Jak můžeme vidět klient při stahování využil maximálně 31.6kbyte/s což je po přepočtení 252,8 kbit/s.

Z naměřeného výsledku usuzujeme, že námi aplikovaný skript HTB je funkční a omezuje rychlost dle potřeby a nastavení. Neboť reálná rychlost sítě je mnohonásobně větší.

7.2.2 Dva klienti

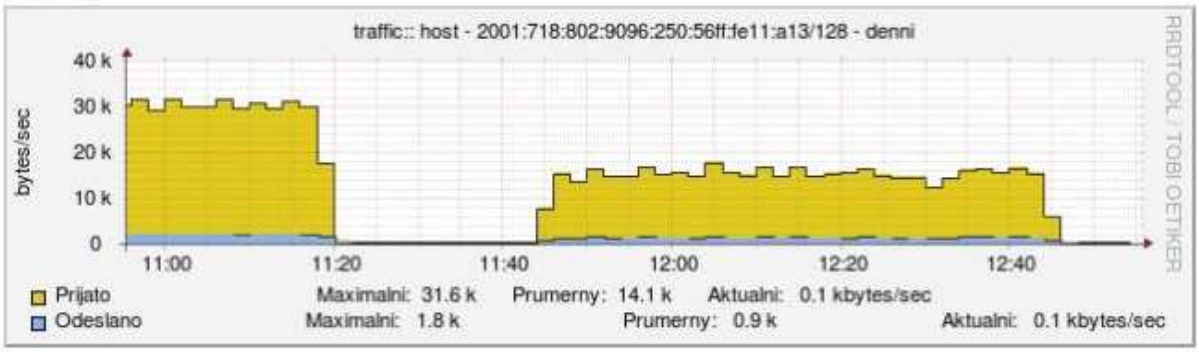
Nastavení hodnoty rate v HTB skriptu ponecháváme na stejné hodnotě jako pro měření stahování jedním klientem, tj. na hodnotě 256kbit/s. Ve skriptu sice nastavujeme hodnotu rate, ale ve skutečnosti se jedná o hodnotu ceil, tedy o rychlost celkovou dané linky.
K ověření dvou klientů jsme měli k dispozici jednoho virtuálního klienta s distribucí Arch Linux a druhého na distribuci Fedora.
Na začátku měření je nutné překontrolovat nastavení route, ověřit konektivity a překontrolovat, zda opravdu veškerá komunikace jde přes námi definovaný server. Oba klienty přihlásíme k ftp6 serveru, který se nachází na našem serveru.
Na obou klientech jsme zahájili současně stahování souboru soubor50Mib z ftp6 serveru.



Obr.13: Graf stahování souboru dvěmi klienty současně – klient Fedora

Na grafu můžeme vidět, že doba stahování souboru na jednom klientovi byla dvojnásobná. Při stahování souboru jedním klientem trvalo klientovi stažení souboru cca 30 minut. Při stahování souboru dvěmi klienty byla doba stahování 6O minut. Na grafu není zaznamenán náběh, neboť RRD začal generovat graf při náběhu provozu. Vhodné by bylo prvně zadat alespoň příkaz ping, aby RRD zaznamenalo klientovu aktivitu. Nicméně tato skutečnost nemění nic na výsledném zobrazení grafu.
Jak z měření vyplývá, klient na distribuci Fedora dosáhl průměrné rychlosti stahovaných dat 13,1 kbyte/s což po přepočtení je 104,8 kbit/s. Klient s distribucí Arch Linux dosáhl průměrné hodnoty 14,1 kbyte/s což je 112,8 kbit/s. Po součtu obou rychlostí dosáhneme hodnoty 217,6 kbit/s. Připočteme-li průměrné hodnoty odeslaných packetů 1,6 kbyte/s dostáváme se v celkovém součtu na hodnotu 230,5 kbit/s. Jelikož vycházíme z průměrných hodnot a

virtuální klient běžel přes Putty, tak myslíme, že dosažená hodnota měření je velmi dobrá potvrzuje správnost nastavení HTB filtru.



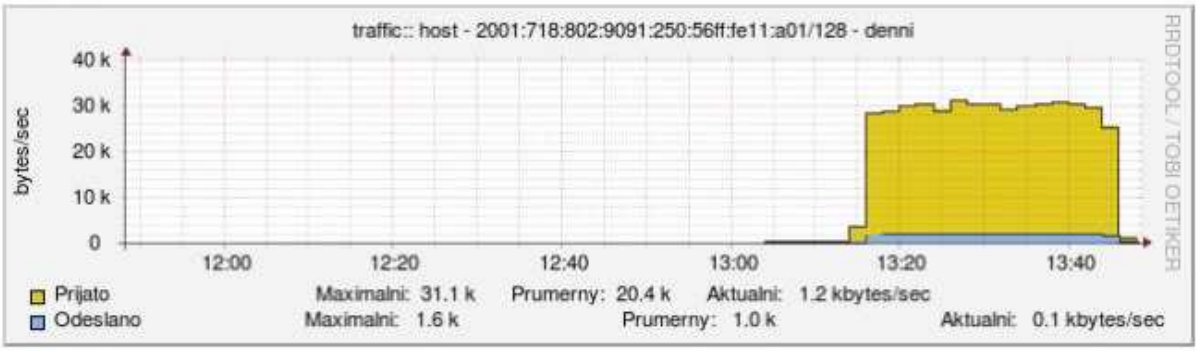
Obr.14: Graf porovnání stahování jedním a dvěmi klienty

Na výše vyobrazeném grafu vidíme, srovnání dobíhající stahování jedním klientem a celé stahování souboru dvěmi klienty. Kdy tento graf vyobrazuje provoz jednoho klienta. Jak je z grafu více než dobře patrné, tak rychlost stahování, nebo-li rychlost přijatých paketů je poloviční.

7.2.3 Čtyři klienti při hodnotě rate 1000 kbit/s

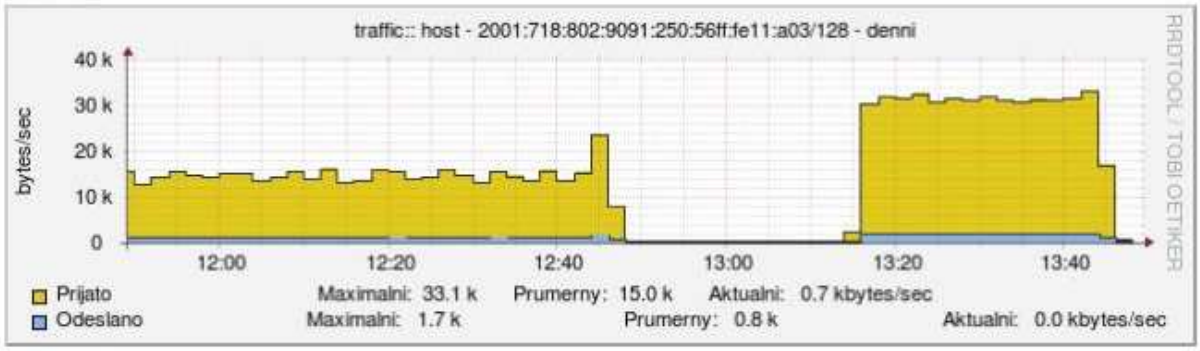
Při měření hodnot čtyř klientů jsme pozměnili nastavení skriptu HTB. Hodnotu RATE jsme nastavili na 1000 kbit/s. Opět nutno dodat, že se ve skutečnosti jedná o hodnotu ceil, tedy o celkovou rychlost linky. K měření nám posloužili tři klienti na distribuci Linux Fedora a jeden klient distribuce Arch Linux.

Opět je zapotřebí ověřit konektivitu se serverem a přihlásit všechny klienty k ftp6 serveru. Po nachystání měření jsme téměř současně na všech klientech spustili stahování souboru soubor50Mib.dat.



Obr.15: Graf stahování čtyř klientů současně při hodnotě rate 1000 kbit/s

Při navýšení hodnoty rate na hodnotu 1000 kbit/s a ponechání ostatních hodnot a nastavení ve skriptu HTB dochází k dělení rychlosti 1000 kbit/s mezi čtyři klienty v daném případě. Opět je nutno podotknout, že daná hodnota je celková rychlost linky, nebo-li ceil. Z grafu měření můžeme vyčíst, že časový interval stažení souboru soubor50Mib.dat se zkrátil opět na 30 minut, což je stejná hodnota jako při stahování souboru jedním klientem při nastavení hodnoty rate na 256 kbit/s. Průměrná rychlost stažení jedním klientem je tedy 163,2 kbit/s.



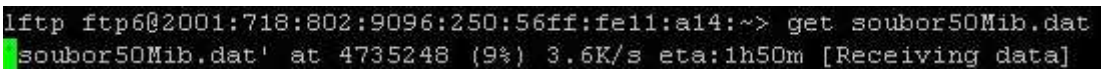
Obr.16: Graf porovnání stahování klienta při různých hodnotách rate

Na grafu porovnání stahování klienta při různých hodnotách rate vidíme výsledky měření při nastavení hodnoty rate na 256 kbit/s a stažení souboru dvěmi klienty. V druhé polovině grafu vidíme stažení souboru soubor50Mib.dat při nastavení hodnoty rate na 1000 kbit/s při zatížení čtyřmi klienty.

Troufáme si říci, že daný výsledek měření je jednoznačně důkaz ovlivnění QoS pomocí HTB skriptu. Maximální hodnota dosažené rychlosti je 33,1kbyte/s což po přepočtení znamená 264,8 kbit/s.

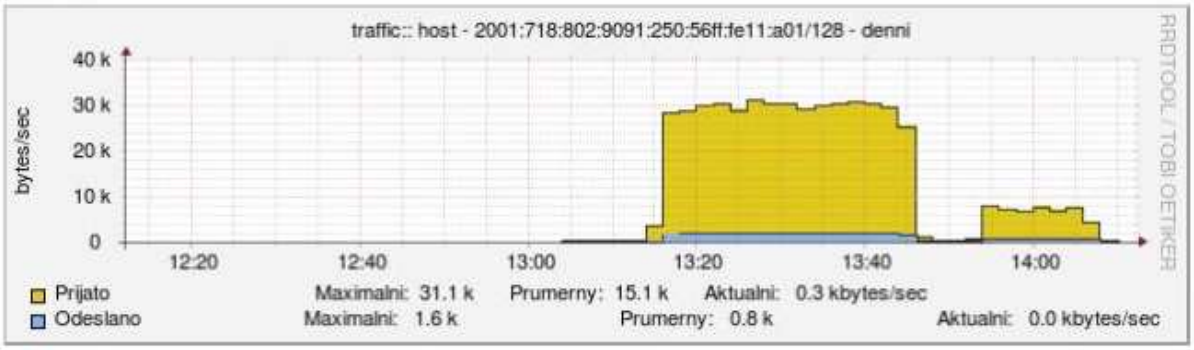
7.2.4 Čtyři a méně klientů při hodnotě rate 256 kbit/s

Pro další měření čtyř klientů jsme se rozhodli nastavit hodnotu RATE na 256 kbit/s, opět se jedná o celkovou rychlost linky. K měření nám posloužili tři klienti na distribuci Linux Fedora a jeden klient distribuce Arch Linux.
Opět je zapotřebí ověřit konektivitu se serverem a přihlásit všechny klienty k ftp6 serveru. Po nachystání měření jsme téměř současně na všech klientech spustili stahování souboru soubor50Mib.dat.



Obr.17: Stahování souboru čtyřmi klienty současně

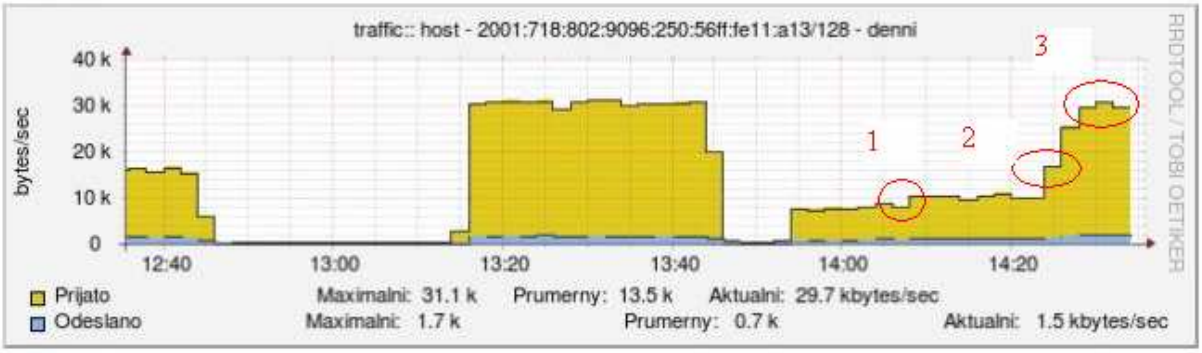
Jak můžeme vidět na obrázku, tak při stahování souboru čtyřmi klienty současně dochází k značnému zpomalení při stahování. Jednak čas stahování pro jednoho klienta Se pohybuje okolo 120 minut a aktuální rychlost jednoho klienta je okolo 28,8 kbit/s což pro dnešní nároky na provoz je naprosto nedostačující.
Pro porovnání jsme v grafu ponechaly hodnoty při stahování souboru čtyřmi klienty při hodnotě rate 1000 kbit/s.



Obr.18: Graf klienta při rate 1000 kbit/s a 256 kbit/s při stahování čtyřmi klienty

Na výše uvedeném grafu vidíme výsledek měření, při stahování souboru soubor50Mib.dat čtyřmi klienty současně. V první části je hodnota rate nastavena na 1000 kbit/s. Ve druhé části je hodnota rate nastavena na 256 kbit/s.

Stažení souboru neproběhlo celé. Klient při nastavení hodnoty rate 256 kbit/s byl od ftp6 odpojen při stažení 10% velikosti souboru. Je patrné, že za čas, který je potřebný ke stažení celého souboru jedním klientem při nastavení hodnoty rate 1000 kbit/s, při nastavení hodnoty rate na 256 kbit/s je za přibližně stejnou dobu staženo pouze 10% souboru.



Obr.19: Graf klienta1 při rate 1000 kbit/s a 256 kbit/s při stahování čtyřmi klienty

Graf klienta1 při rate 1000 kbit/s a 256 kbit/s při stahování čtyřmi klienty nám názorně ukazuje průběh celého měření. První třetina obsahuje graf měření pro dva klienty při rate 256 kbit/s. Prostřední část grafu nám ukazuje měření pro čtyři klienty pro rate 1000 kbit/s. Poslední třetina grafu zobrazuje měření při hodnotě rate 256 kbit/s pro čtyři klienty.

Jelikož jsme naznali, že nemá cenu měřit celou dobu stahování, kdy se objeví jen minimální odchylky, tak jsme se rozhodli, že měření rozdělíme do tří částí. Jednotlivé přechody jsme na grafu označili číslicemi 1,2,3 a oblast zájmu označili.

V první části jsme měli puštěny čtyři klienty, kteří stahovali soubor50Mib.dat současně. V námi označené první oblasti zájmu jsme vypnuli jednoho klienta. Bylo to při deseti procentech staženého souboru u každého klienta. Tzn. od oblasti zájmu 1 po oblast zájmu 2 stahovali tři klienti při nastavení hodnoty rate na 256 kbit/s. Jak je z grafu patrné, tak víceméně se žádné velké navýšení rychlosti stahování jednotlivých klientů nekonalo.

Oblast zájmu 2 označuje místo vypnutí dalších dvou klientů od ftp6. Zde již je patrné navýšení rychlosti poslednímu klientovi. Ten postupně svoji rychlost přidává, až se dostane na označení zájmu 3, kde klient obsadí plnou šířku přiděleného pásma a stahuje rychlostí 256 kbit/s.

8. Závěr

Cílem diplomové práce bylo zjistit teoreticky současné možnosti QoS na verzi IPv6 protokolu a získané poznatky ověřit v praxi. Daná problematika je implementována pro Linux. V teoretické části jsme vycházeli z faktu, že teoretické podklady pro sítě jako celek se nevztahují pro jednotlivé operační systémy. Daná specifikace byla začleněna při samotném vypracování a realizaci sítě.

V teoretické části jsme se seznámil s teoretickými podklady. Dané studium zahrnulo jednotlivé verze Linuxu, prokoly IPv4 a IPv6, dále specifikaci QoS. Seznámili jsem se také s jednotlivým řízením front, klasifikací a zpracováním paketů.

V praktické části popisujeme, jak jsme síť pod IPv6 sestavovali od instalace jednotlivých stanic a serveru až k její kompletní funkčnosti. Původním záměrem bylo zprovoznit server pod distribucí Linux SUSE. Na dané distribuci se nám bohužel nepodařilo zprovoznit plně forwarding. Obdobně jsme dopadli i s distribucí Linux Fedora. Dané distribuce jsou už příliš upraveny a podobná Windows a neumožňují správci plnou konfiguraci systému dle jeho představ.

Dalším krokem bylo zprovoznění skriptu HTB, který zpracovává provoz pomocí nastavených pravidel. Pomocí aplikace ipaccounting a RRD jsme zajistili, že veškerý provoz je značkován a sledován. Výsledky jsou zaznamenávány v grafech.

V poslední části diplomové práce jsme rozebrali jednotlivé grafy a výstupy z měření. Měření jsme prováděli pro nastavení hodnoty rate na 256 kbit/s a pro hodnotu 1000 kbit/s. Jednotlivé měření jsme oddělili počtem klientů, kteří byli připojeni k serveru.

Literatura

[1] NEMETH, E., SNYDER, G., HEIN T. Linux - Kompletní příručka administrátora. Computer Press, 2004. 880 s. ISBN: 80-722-6919-4.

[2] HAGEN, S. IPv6 Essentials. O' Reilly Media, 2006. 407s. ISBN: 0-596-10058-2.

[3] Sven Ubik,
Technická zpráva TEN-155 CZ číslo 6/2000
<http://staff.cesnet.cz/~ubik/publications/2000/diffserv.html>

[4] Bigelow, Stephen J.,
Mistrovství v počítačových sítích: správa konfigurace, diagnostika a řešení problémů Computer press Brno 2004

[5] *Eldar* [online]. 2000 [cit. 2009-12-01]. Dostupný z WWW:
<http://eldar.cz/manasek/felbox/36mps/qos/index.htm>.

[6] *Svet sití* [online]. 2000 [cit. 2000-12-01]. Dostupný z WWW:
<http://www.svetsiti.cz/view.asp?rubrika=Tutorialy&temaID=70&clanekID=75>.

[7] *Linux* [online]. 2000 [cit. 2009-12-01]. Dostupný z WWW:
<http://www.linux.cz/distribuce.html>.

[8] *Cesnet* [online]. 2000 [cit. 2009-12-01]. Dostupný z WWW:
<http://www.cesnet.cz/qos/>.

[9] MATUŠINA, František. *Specifikace databáze MIB pro technologii DiffServ*. [s.l.], 2007. 85 s. Bakalářská práce.

[10] *Wiki.archlinux.org* [online]. 2000 [cit. 2010-05-15]. Wiki.archlinux.org. Dostupné z WWW:
<http://wiki.archlinux.org/index.php/Official_Arch_Linux_Install_Guide_%28%C4%8Cesky%29>.

[11] *Www.logix.cz* [online]. 2000 [cit. 2010-05-15]. IPv6 krok za krokem. Dostupné z WWW : <http://www.logix.cz/michal/doc/article.xp/ipv6-1>.

[12] *Http://tldp.org* [online]. 2000 [cit. 2010-05-15]. Linux IPv6 HOWTO. Dostupné z WWW: <http://tldp.org/HOWTO/Linux+IPv6-HOWTO/>.

[13] *Www.root.cz* [online]. 2000 [cit. 2010-05-15]. HTB jemný úvod. Dostupné z WWW: <http://www.root.cz/clanky/htb-jemny-uvod/>.

[14] *Www.ipaccounting.standus.com* [online]. 2000 [cit. 2010-05-15]. IP Accountinf. Dostupné z WWW: <http://ipaccounting.standus.com/>.

[15] *Ashus* [online]. 2000 [cit. 2010-05-15]. Ashus. Dostupné z WWW: <http://ashus.ashus.net/viewtopic.php?f=16&t=66>.

[16] *Http://oss.oetiker.ch* [online]. 2000 [cit. 2010-05-15]. RRDtool - About RRDtool. Dostupné z WWW: <http://oss.oetiker.ch/rrdtool/>.

[17] *Http://tldp.org* [online]. 2000 [cit. 2010-05-15]. FTP mini-HOWTO. Dostupné z WWW: <http://tldp.org/HOWTO/FTP-6.html>.

[18] HOLÍK, Aleš. *Kontrola rychlosti přenosu dat*. Zlín, 2007. 55 s. Bakalářská práce. Univerzita Tomáši Bati ve Zlíně.

Přílohy

HTB.sh

```
RATE=256
tc qdisc del dev eth0 root
tc qdisc add dev eth0 root handle 1:0 htb default 14
#Vymazani soucasnych qdiscu

tc class add dev eth0 parent 1:0 classid 1:1 htb rate ${RATE}kbit
tc class add dev eth0 parent 1:1 classid 1:11 htb rate $[${RATE}/4]kbit ceil ${RATE}kbit
tc class add dev eth0 parent 1:1 classid 1:12 htb rate $[${RATE}/4]kbit ceil ${RATE}kbit
tc class add dev eth0 parent 1:1 classid 1:13 htb rate $[${RATE}/4]kbit ceil ${RATE}kbit
tc class add dev eth0 parent 1:1 classid 1:14 htb rate $[${RATE}/4]kbit ceil ${RATE}kbit
#Hlavni trida ma rychlost 256 a ostatni uzly 64

tc qdisc add dev eth0 parent 1:11 handle 11:0 sfq perturb 10
tc qdisc add dev eth0 parent 1:12 handle 12:0 sfq perturb 10
tc qdisc add dev eth0 parent 1:13 handle 13:0 sfq perturb 10
tc qdisc add dev eth0 parent 1:14 handle 14:0 sfq perturb 10
#Na konci kazde vetve je SFQ prepocitavana co 10sek

ip6tables -t mangle -F FORWARD
ip6tables -t mangle -A FORWARD -j MARK --set-mark 4
ip6tables -t mangle -A FORWARD -d 2001:718:802:9091:250:56ff:fe11:a13 -j MARK --set-
mark 1
ip6tables -t mangle -A FORWARD -d 2001:718:802:9091:250:56ff:fe11:a10 -j MARK --set-
mark 2
ip6tables -t mangle -A FORWARD -d 2001:718:802:9091:250:56ff:fe11:a9 -j MARK --set-
mark 3
ip6tables -t mangle -F OUTPUT
ip6tables -t mangle -A OUTPUT -p tcp --sport 3128 -j MARK --set-mark 4

ip6tables -t mangle -A OUTPUT -p tcp --sport 3128 -d 2001:718:802:9091:250:56ff:fe11:a13
-j MARK --set-mark 1
ip6tables -t mangle -A OUTPUT -p tcp --sport 3128 -d 2001:718:802:9091:250:56ff:fe11:a10
-j MARK --set-mark 2
ip6tables -t mangle -A OUTPUT -p tcp --sport 3128 -d 2001:718:802:9091:250:56ff:fe11:a9 -
j MARK --set-mark 3
#Doplnit ip adresy jednotlivých klientů

tc filter add dev eth0 parent 1:0 protocol ip handle 1 fw flowid 1:11
tc filter add dev eth0 parent 1:0 protocol ip handle 2 fw flowid 1:12
tc filter add dev eth0 parent 1:0 protocol ip handle 3 fw flowid 1:13
tc filter add dev eth0 parent 1:0 protocol ip handle 4 fw flowid 1:14
```

Ipaccounting.sh

```
#!/bin/bash
#
#  IP Accounting v 1.1
#  Tento program slouzi ke statistice sitoveho provozu pro jednotliva IP
#  Created by standus - standus@standus.com
#  Upraveno by František Matusina pro IPv6

active_iface="eth0"
dir_data_txt="/var/www/html/ipaccounting"
dir_data_rdd="/var/www/html/ipaccounting/rrd"
rdd_active="on"
txt_active="on"
testing_active="on"

# Testovani pravidel a chainu v IP6TABLES
if [ $testing_active == "on" ]; then
   if [ "`ip6tables -t mangle -L POSTROUTING | grep "DATA_IN"`" == "" ]; then
      echo "Pridavam chain DATA_IN"
      ip6tables -t mangle -N DATA_IN
      ip6tables -t mangle -A POSTROUTING -j DATA_IN
   fi
   if [ "`ip6tables -t mangle -L PREROUTING | grep "DATA_OUT"`" == "" ]; then
      echo "Pridavam chain DATA_OUT"
      ip6tables -t mangle -N DATA_OUT
      ip6tables -t mangle -A PREROUTING -j DATA_OUT
   fi

   for iface in $active_iface; do
#      for ip in `cat /proc/net/arp | grep "$iface" | awk '{print $1}'`; do
      for ip in `ip -6 neigh show dev "$iface" | awk '{print $1}'`; do
         echo "$2 Testuji ip: $ip"
#echo "ip6tables -t mangle -L DATA_IN  -v -x -n | grep " $ip " | awk '{print $2}'"
         if [ "`ip6tables -t mangle -L DATA_IN  -v -x -n | grep "$ip" | awk '{print $2}'`" == ""
]; then
            echo "Pridavam pravidlo DATA_IN pro ip: $ip"
            ip6tables -t mangle -A DATA_IN -d $ip -j RETURN
         fi
         if [ "`ip6tables -t mangle -L DATA_OUT  -v -x -n | grep "$ip" | awk '{print $2}'`" ==
"" ]; then
            echo "Pridavam pravidlo DATA_OUT pro ip: $ip"
            ip6tables -t mangle -A DATA_OUT -s $ip -j RETURN
         fi
      done
   done
fi

# Nacteni dat z IP6TABLES
counter_file=`cat $dir_data_txt/data.txt`
```

```
data=`ip6tables -t mangle -L DATA_OUT  -v -x -n -Z  | sed -e 's/ /-/g' | grep "RETURN"`
for i in $data; do
   ip=`echo $i | sed -e 's/-/ /g' | awk '{print $7}'`
   out=`echo $i | sed -e 's/-/ /g' | awk '{print $2}'`
   out_old=`echo -ne "$counter_file"  | grep "$ip-" | sed -e 's/-/ /g' | awk '{print $2}'`
   out_new=$(($out_old + $out))
   new_file_out="$new_file_out$ip-$out_new\n"
   out_aver=$(($out / 120))
   new_rdd_out="$new_rdd_out$ip-$out_aver\n"
   echo -ne "$ip\tOUT\tstare: $out_old \tnove: $out \tsoucet: $out_new\trate: $out_aver\n"
done
data=`ip6tables -t mangle -L DATA_IN  -v -x -n -Z  | sed -e 's/ /-/g' | grep "RETURN"`
for i in $data; do
   ip=`echo $i | sed -e 's/-/ /g' | awk '{print $8}'`
   in=`echo $i | sed -e 's/-/ /g' | awk '{print $2}'`
   in_old=`echo -ne "$counter_file"  | grep "$ip-" | sed -e 's/-/ /g' | awk '{print $3}'`
   in_new=$(($in_old + $in))
   new_file_in="$new_file_in$ip-$in_new\n"
   in_aver=$(($in / 120))
   new_rdd_in="$new_rdd_in$ip-$in_aver\n"
   echo -ne "$ip\tIN\tstare: $in_old \tnove: $in \tsoucet: $in_new\trate: $in_aver\n"
done



# Zapsani hodnot do txt souboru (pocitadlo prenesenych dat)
if [ $txt_active == "on" ]; then
   for i in `echo -ne $new_file_out`; do
      ip=`echo $i | sed -e 's/-/ /g' | awk '{print $1}'`
      out=`echo $i | sed -e 's/-/ /g' | awk '{print $2}'`
      in=`echo -ne "$new_file_in" | grep "$ip-" | sed -e 's/-/ /g' | awk '{print $2}'`
      counter_data="$counter_data$ip-$out-$in\n"
   done
   echo -ne $counter_data > $dir_data_txt/data.txt
fi

# Zapsani hodnot do rrd databaze (grafy prenosu)
if [ $rdd_active == "on" ]; then
   for i in `echo -ne $new_rdd_out`; do
      ip=`echo $i | sed -e 's/-/ /g' | awk '{print $1}'`
      fip=`echo $ip | sed -e "{s/\:/\_/g}" | sed -e "{s/\//\x/g}"`
      out=`echo $i | sed -e 's/-/ /g' | awk '{print $2}'`
      in=`echo -ne "$new_rdd_in" | grep "$ip-" | sed -e 's/-/ /g' | awk '{print $2}'`
         if [ ! -e "$dir_data_rdd/host-$fip.rrd" ]; then
            rrdtool create "$dir_data_rdd/host-$fip.rrd" --step 120 DS:in:GAUGE:600:0:U
DS:out:GAUGE:600:0:U RRA:AVERAGE:0.5:1:3600 RRA:AVERAGE:0.5:6:3600
RRA:AVERAGE:0.5:42:3600;
         fi
         rrdtool update "$dir_data_rdd/host-$fip.rrd" -t in:out N:$in:$out
   done
fi
```