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

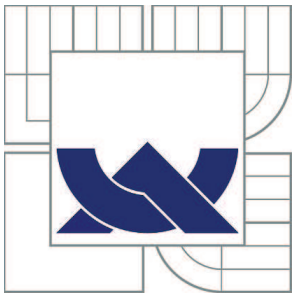
IMPLEMENTACE POKROČILÉ FILTRACE S KLASIFIKACÍ PAKETŮ PRO BEZDRÁTOVÉ SÍTĚ

DIPLOMOVÁ PRÁCE
MASTER'S THESIS

AUTOR PRÁCE
AUTHOR

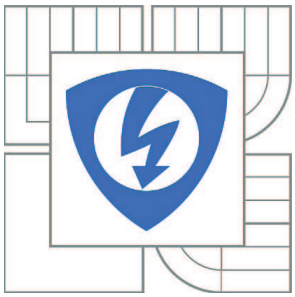
Bc. MILAN GRÉNAR

BRNO 2011



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

IMPLEMENTACE POKROČILÉ FILTRACE S KLASIFIKACÍ PAKETŮ PRO BEZDRÁTOVÉ SÍTĚ

IMPLEMENTATION OF ADVANCED FILTRATION WITH THE CLASSIFICATION OF PACKETS FOR A WIRELESS NETWORK

DIPLOMOVÁ PRÁCE

MASTER'S THESIS

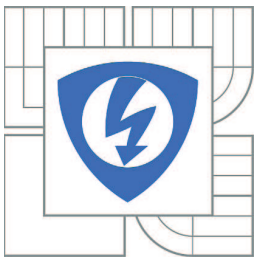
AUTOR PRÁCE — Bc. MILAN GRÉNAR

AUTHOR

VEDOUCÍ PRÁCE — Ing. JURAJ SZŐCS

SUPERVISOR

BRNO 2011



VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ

Fakulta elektrotechniky a komunikačních technologií

Ústav telekomunikací

Diplomová práce

magisterský navazující studijní obor
Telekomunikační a informační technika

Student:	Bc. Milan Grénar	***ID:***	73038
Ročník:	2	***Akademický rok:***	2010/2011

NÁZEV TÉMATU:

Implementace pokročilé filtrace s klasifikací paketů pro bezdrátové sítě

POKYNY PRO VYPRACOVÁNÍ:

Prozkoumejte možnosti SW řízení QoS v prostředí OS Linux. Vyberte si dvě podle Vás nejvhodnější metody pro řízení QoS v prostředí OS Linux a zdůvodněte, proč jste je vybral. Vámi vybrané metody aplikujte v reálné bezdrátové síti 802.11 a porovnejte jejich výhody a nevýhody se standardem 802.11e. Součástí práce bude taky simulace bezdrátové sítě v simulačním prostředí NCTUns, která bude využívat standard 802.11e. Analyzujte efektivitu zajištění QoS při datech citlivých na zpoždění.

DOPORUČENÁ LITERATURA:

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

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

Vedoucí práce: Ing. Juraj Szőcs

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

UPOZORNĚNÍ:

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

ABSTRAKT

Práce se zabývá možnostmi řízení QoS v GNU/Linux nástroji iptables a iproute. Pozornost je zaměřena zejména na metody pro tvarování provozu HTB a HFSC s ohledem na nasazení v bezdrátových sítích. Součástí je také simulace zajištění QoS využívající doplněk 802.11e.

KLÍČOVÁ SLOVA

Iptables, HTB, HFSC, IEEE 802.11e, QoS, Traffic Control, WME, WMM

ABSTRACT

The diploma thesis addresses facility of QoS control with GNU/Linux tools iptables and iproute. An attention is focused especially on HTB and HFSC traffic shaping methods with regard to utilization in wireless networks. The paper also includes a simulation of ensuring QoS in wireless network with 802.11e amendment.

KEYWORDS

Iptables, HTB, HFSC, IEEE 802.11e, QoS, Traffic Control, WME, WMM

GRÉNAR, Milan *Implementace pokročilé filtrace s klasifikací paketů pro bezdrátové sítě*: diplomová práce. Brno: Vysoké učení technické v Brně, Fakulta elektrotechniky a komunikačních technologií, Ústav telekomunikací, 2011. 61 s. Vedoucí práce byl Ing. Juraj Szőcs

PROHLÁŠENÍ

Prohlašuji, že svou diplomovou práci na téma „Implementace pokročilé filtrace s klasifikací paketů pro bezdrátové sítě“ 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.

Brno ....

(podpis autora)

OBSAH

SEZNAM OBRÁZKŮ

SEZNAM TABULEK

ÚVOD

Diplomová práce se věnuje mechanismům pro tvarování provozu v aktivních síťových prvcích běžících na GNU/Linux. Cílem práce je zejména prozkoumání vybraných metod vhodných pro přidělování šířky pásma a zajištění kvality služeb jednotlivým uživatelům či jejich aplikacím. Tyto metody využívají především poskytovatelé internetu, kteří firmám zaručují požadované garantované toky a maximální zpoždění. V souvislosti s tvarovači provozu jsou popsána specifika pro jejich nasazení na bezdrátové sítě.

Záměrem praktické realizace je ověření záruky QoS na bezdrátové síti spolu s koexistencí nástrojů pro tvarování provozu. První část se zabývá simulací chování doplňku 802.11e v programu NCTUns. Druhá část zavádí tvarovače HTB a HFSC do reálného prostředí s WMM certifikovanou bezdrátovou síťovou kartou.

1 ŘEŠENÍ STUDENTSKÉ PRÁCE

1.1 Teoretický úvod

Při přenosu dat pomocí sítě internet je s běžnými pakety ve většině případů zacházeno způsobem *best effort*, což znamená, že síť se sice snaží o úspěšné doručení paketů, nicméně nezaručuje dosažení jejich cíle ani žádnou kvalitu přenosu. Toto schéma však nevyhovuje u aplikací, které jsou citlivé na šířku pásma přenosové trasy, na zpoždění, kolísání zpoždění nebo zahazování paketů. Jedná se například o přenos hlasu nebo obrazu, kdy je potřeba zajistit odlišení datového toku od ostatního provozu a jiné prioritní zacházení na uzlech sítě. Právě k tomuto účelu slouží kvalita služeb – *Quality Of Service*. Parametry používané u QoS v souvislosti s multimédii jsou:

- ztrátovost *loss*,
- zpoždění *delay*,
- kolísání zpoždění *jitter*,
- propustnost *troughput*.

Existují dvě základní architektury, zajišťující rozpoznání typu datových jednotek v síti a jim odpovídající způsob zacházení, a to integrované služby – *Integrated Services* (IntServ) a diferencované služby – *Differentiated Services* (DiffServ).

1.1.1 IntServ

Vznik IntServ (RFC 1633) byl podmíněn požadavky real-time aplikací. Klíčovými mechanismy této architektury jsou *Admission Control* a protokol pro rezervaci síťových prostředků *Resource Reservation Protocol*. AC nejdříve rozhodne, zda je síť schopna uspokojit požadavky koncových bodů. Pokud ano, RSVP sjedná požadované parametry na všech směrovačích a přepínačích podél celé trasy přenosu. Síťové prvky následně udržují informace, přidělené každému spojení. Tento přístup sice zaručuje parametry pro konkrétní datové toky, ale způsobuje vyšší časovou režii a menší škálovatelnost. Proto je v dnešní době nejpoužívanější DiffServ.

1.1.2 DiffServ

Architektura rozděluje tok datových jednotek do tříd a těmto třídám garantuje specifické zacházení. V současnosti jde o nejpoužívanější metodu. Principem této metody je rozdělení paketů do tříd podle jejich vlastností při příchodu na směrovač. S třídami je zacházeno podle definovaného způsobu tzv. *per router behaviour*.

1.2 Možnosti řízení QoS na Linuxu

Termín QoS je často používán jako synonymum pro řízení provozu. Řízení provozu v podstatě umožňuje uživateli konfigurovat soupravu systémů a mechanismů front, kterým jsou pakety přijímány a vysílány síťovým zařízením. Během zpracování může docházet k rozhodování, zda a které pakety přijímat, jakou rychlostí na vstupu rozhraní a které pakety vysílat a v jakém pořadí na výstupu.

Vhodně nastavené řízení provozu by mělo vést především k předvídatelnějšímu využití síťových zdrojů. Reálně se může jednat o omezení celkové dostupné šířky pásma na určitou rychlost; omezení šířky pásma nebo její rezervaci pro konkrétního uživatele, službu nebo klienta; upřednostnění provozu citlivého na zpoždění nebo spravedlivé rozdělení nerezervované šířky pásma. Existuje tedy celá řada možností použití, uvedený výčet tak zdaleka není úplný. Pro konkrétní případy je také nutné zvolit příslušné komponenty (mechanismy front, filtry atp.).

Klasické prvky řízení provozu

Značkování *marking* - obecně provádění změny v paketu - poskytuje značku, která je využita klasifikací. Značkovací mechanismus nastavuje v hlavičce paketu určitý DSCP, což jej přiřadí do skupiny paketů se stejným chováním.
Klasifikace *classifying* - roztřídění či separace paketů do front podle obsahu jejich hlavičky v souladu s definovanými pravidly. Může se jednat o tzv. klasifikaci

- *Multi Field* (MF) - zkoumání jednoho nebo více polí v hlavičce jako je zdrojová či cílová IP adresa, port nebo protokol.
- *Behavior Aggregate* (BA), zaměřující se pouze na značku v šestibitovém poli *Differentiated Service Code Point* (DSCP), dva další bity *Currently Unused* (CU) zůstávají nevyužity. DSCP dovoluje každému směrovači v cestě určit, jak by měla být každá třída provozu klasifikována. DSCP a CU se nachází v DS poli (viz tab. 1.1). U IPv4 je pole uloženo v místě ToS, které původně sloužilo pro jiné účely, než je podpora technologie DiffServ. Namísto absolutní priority IP paketu u ToS udává DSCP identifikátor třídy, nastavitelný administrátorem v jedné DiffServ doméně[1].

Tvarování *shaping* - zpožďování paketů za účelem dosažení přenosové rychlosti pod stanovenou úrovní. Základním mechanismem pro tvarování na určitou rychlost je mechanismus Token Bucket.
Zahazování *dropping* - zahození celého paketu, toku nebo klasifikace.
Plánování *scheduling* - skládání či přeskládávání paketů na výstup.
Dohlížení *policing* - měření a omezení provozu v konkrétní frontě.

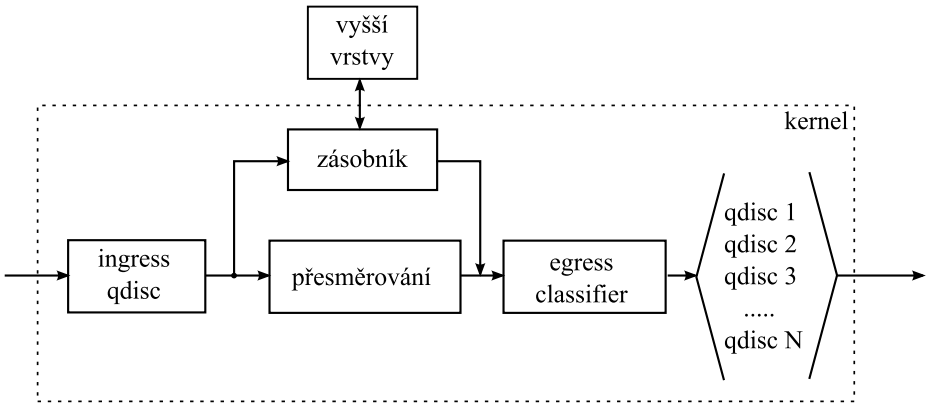
[1] u IPv6 je DiffServ pole uloženo v položce Traffic Class.

Tab. 1.1: ToS bity v hlavičce paketu IPv4

bity	0	1	2	3	4	5	6	7
DS pole	DSCP						CU	
ToS pole	PRECEDENCE			ToS				MBZ

Řízení provozu v Linuxu je možno zajistit nástroji traffic control a iptables. Obr. 1.1 znázorňuje kontrolu provozu na Linuxu využitím těchto nástrojů a jejich součástí, jejichž konkrétnější popis bude následovat.

Pakety přicházejí přes vstupní rozhraní vlevo. V bloku ingress qdisc mohou být aplikovány filtry, které rozhodnou o případném zahození nežádoucích paketů, například pokud nastane překročení určité rychlosti vstupních dat. Zahozením paketů ihned na vstupu předcházíme zbytečnému zatěžování procesoru. Poté jsou povolené pakety buďto poslány přes zásobník k vyšším vrstvám lokální aplikace, která může také generovat data, nebo dojde k přesměrování - *forwardingu*. Blok přesměrování zahrnuje výběr výstupního rozhraní, next hopu, atp. Dále data vstoupí do jedné z qdisc na definovaném rozhraní. Zde hraje velkou roli traffic control, který rozhoduje o vstupu paketů do fronty nebo jejich zahození; pořadí či zpoždění odeslání[2]. Jakmile je paket jádrem uvolněn pro odeslání, ovladač ho zpracuje a vyšle na síť.[4]



Obr. 1.1: Zpracování paketů na linuxovém směrovači

[2]ve výchozím nastavení je použita pouze jedna egress qdisc - pfifo_fast.

1.2.1 Iptables

Pro filtraci IPv4 paketů, vybudování a údržbu NAT a zejména modifikaci položek v těle hlavičky v linuxu slouží balíček netfilter. V souvislosti s QoS se může jednat o značkování. Nástrojem obvykle spojovaným s netfilter pro linuxová jádra 2.4.x a 2.6.x je iptables.

Iptables má za úkol zprostředkovat jádru záměry uživatele s příchozími, odchozími či procházejícími pakety přes rozhraní linuxu. Hlavními používanými entitami jsou tabulky, obsahující volitelný počet řetězců - seznamů pravidel. Každé pravidlo specifikuje určité filtrovací parametry, např. rozhraní, protokoly nebo porty. Pokud je testovaný paket v souladu s těmito parametry, provede se nad ním předem definovaná akce. Následující příkaz ukazuje přidání pravidla do řetězce CHAIN s akcí TARGET.

```
iptables  -A CHAIN <specifikace filtrovacích parametrů> -j TARGET
```

Výchozí tabulka *filter* obsahuje tři řetězce: INPUT, FORWARD a OUTPUT. Další tabulky a řetězce lze volitelně doplňovat. Načtením NAT a *mangle* modulů získáme tabulky NAT a *mangle*. Pomocí druhé vyjmenované lze modifikovat ToS byte nebo značkovat pakety číslicí explicitně pomocí přepínače `--set mark` *`číslo`*.

1.2.2 Traffic Control

Traffic control - tc je důmyslný systém pro řízení provozu na Linuxu, schopný plně zastoupit drahé vyhrazené QoS zařízení. Spolu s nástrojem ip (pro nastavení rozhraní, ARP, tunelování, politiky směrování, atp.) je traffic control obsažen v softwarovém balíčku iproute2, jehož autorem je Alexey Kuznetsov[3]. V tab. 1.2 je uvedena souvislost komponentů tc s klasickými prvky řízení provozu.

Tab. 1.2: Souvislost mezi prvky řízení provozu a komponenty Linuxu

Prvek řízení provozu	**komponent v Linuxu**
tvarování	**class**
plánování	**qdisc**
klasifikace	**filter** prostřednictvím svého nástroje **classifier**
dohlížení	**filter** prostřednictvím svého nástroje **policer**
zahazování	**filter** prostřednictvím svého nástroje **policer**
značkování	nastavení argumentu *dsmark* u qdisc

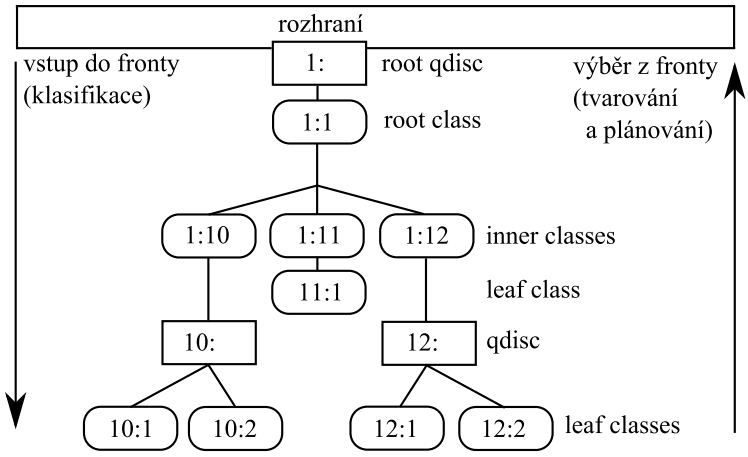
[3]iproute2 poskytuje škálu prostředků pro pokročilé směrování, tunelování a řízení provozu.

1.2.3 Komponenty nástroje Traffic Control

Qdisc

Qdisc *queueing discipline* je hlavním stavebním blokem řízení provozu na Linuxu a zároveň algoritmem, který řídí frontu zařízení. Pomocí qdisc s využitím klasifikátoru lze rozhodnout, které pakety upřednostnit k vyslání před jinými[4]. Hlavními dvěma typy jsou:

- classless qdiscs - nemohou obsahovat třídy, a proto k nim není možné připojit filtry. Protože neobsahují žádnou třídu jako potomka, není zde možnost uplatnění klasifikace.
- classful qdiscs - mohou obsahovat třídy, ze kterých je zpravidla tvořena celá hierarchie. Dále poskytují tzv. handle - identifikátory, ke kterým se připojují filtry. Použití classful qdisc bez tříd není zakázáno, ale nepřináší žádný užitek a navíc se zvyšuje výpočetní náročnost.



Obr. 1.2: Příklad egress hierarchie classful qdisc s komponenty tc

Místa, na které se qdisc v rozhraní připojuje, se pro odchozí provoz označuje termínem egress a pro příchozí provoz termínem ingress. Ingress qdisc umožňuje pouze zahazování nadměrného provozu[5], popřípadě využít virtuální rozhraní IMQ či IFB, o kterých bude pojednáno později v praktické části. Častěji používaným je tedy egress, známý též jako root. Může obsahovat jakoukoliv qdisc s případnými třídami a strukturami tříd.

[4]provádí tedy plánování, proto se označuje jako plánovač.

[5]na ingress qdisc není možno vytvořit třídu, funguje pouze jako objekt, na který lze připojit filtr spolu s policerem.

Příkaz tc pro konfiguraci qdisc vypadá následovně:

```
tc qdisc [ add | del | replace | change | get ] dev STRING
         [ handle QHANDLE ] [ root | ingress | parent CLASSID ]
         [ estimator INTERVAL TIME_CONSTANT ]
         [ [ QDISC_KIND ] [ help | OPTIONS ] ]

       tc qdisc show [ dev STRING ] [ingress]
Kde:
QDISC_KIND := { [p|b]fifo | tbf | prio | cbq | red | etc. }
```

Class

Třída - class je flexibilním prvkem, vyskytujícím se pouze uvnitř classful qdisc. Do třídy lze umístit množství dalších dceřiných tříd nebo jednu qdisc. Pokud je jejím potomkem opět classful qdisc, řízení provozu lze nastavit do větších detailů za cenu náročnější konfigurace. Každá třída může mít také libovolný počet připojených filtrů, dovolujících výběr dceřiných tříd nebo použití filtru k překlasifikování či zahození provozu, vstupujícího do konkrétní třídy. Poznamenejme, že třída zpravidla uložení paketů nechává na qdisc. Existují tři typy tříd:

1. root classes - kořenové:
   - ve stromu vždy maximálně jedna, připojena na root classful qdisc,
   - obsahuje dceřiné třídy.
2. inner classes - vnitřní:
   - obsahují dceřiné třídy,
   - v hierarchii se nachází nad leaf classes,
   - na root může být nahlíženo jako na vnitřní.
3. leaf classes - listové:
   - jsou konečnými třídami v hierarchii,
   - nemohou nikdy obsahovat další dceřiné třídy,
   - obsahují classless qdisc (implicitně FIFO, lze však změnit),
   - pouze listové třídy mohou obsahovat pakety ve své interní frontě.

Příkaz tc pro konfiguraci třídy:

```
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. }
```

Handle

Každá třída a qdisc v hierarchii potřebuje unikátní identifikátor - handle, který je uváděn jako cíl v argumentech `classid` a `flowid` příkazu `tc filter`. Handle sestává ze dvou čísel major a minor, oddělených dvojtečkou[6]. Pro číslování platí následující pravidla:

- všechny objekty v hierarchii řízení provozu se stejným rodičem musejí sdílet stejné major číslo,
- pokud je minor 0, jde o jednoznačnou identifikaci objektu jako qdsic, jakékoliv jiné hodnoty definují objekt jako třídu,
- všechny třídy sdílející jednoho rodiče musejí mít unikátní minor číslo,
- speciální identifikátor ffff:0 je rezervován pro ingress qdisc,
- objekty připojené přímo k root qdiscu se zpravidla značí jedničkou.

Ostatní číslování je v režii uživatele.[6]

Filter

Klasifikaci, nebo-li přiřazení přicházejících paketů do jedné z dceřiných tříd, lze provádět jedním i více filtry, které jsou připojovány buďto na classful qdiscs nebo na třídy. Pakety však nejdříve prochází root qdiscem. Poté, co filtrem na root qdiscu projde paket, může být přesměrován na nějakou z vnitřních či listových tříd, které mohou mít také vlastní filtry a podstoupit další klasifikaci.
Součástmi filtru jsou:

- **Klasifikátor** rozpoznává vlastnosti paketu. Existuje několik klasifikátorů, např.:
  - **u32** rozhoduje pole uvnitř paketu jako jsou zdrojová a cílová IP adresa,
  - **fw** rozhoduje značka, kterou dal paketu firewall,
  - **route** rozhoduje cesta, kterou bude paket směrován.
- **Policer** provádí volání určitou akci nad, a jinou pod stanovenou přenosovou rychlostí. Touto akcí je zpravidla limitování paketů vstupujících do fronty nebo zahození celého provozu splňující konkrétní zvolená kritéria. Každý policer připojený k filtru by měl mít explicitní možnost zahodit pakety pomocí akce **drop**.

  Ačkoli jsou policing a shaping základními prvky řízení provozu pro omezení šířky pásma, použití policeru nikdy nezpozdí provoz, dojít může pouze k zahození. Důvodem je skutečnost, že na ingress qdisc není možno vytvořit třídu.

Definice filtru musí obsahovat argument pro výběr klasifikátoru (nejčastěji u32) a může obsahovat argument pro výběr policeru.[6]

[6]handle jsou externími identifikátory objektů, které používají aplikace. Jádro si udržuje vlastní interní identifikátory pro každý objekt.

Příkaz tc pro konfiguraci filtru je uveden následovně:

```
Usage: 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
```

1.2.4 Classless qdisc

Classless qdiscs nám nedovolují datový tok rozčleňovat do větších detailů, protože nemohou obsahovat třídy spolu s filtry. V linuxu existuje mnoho implementací. Častěji používanými jsou:

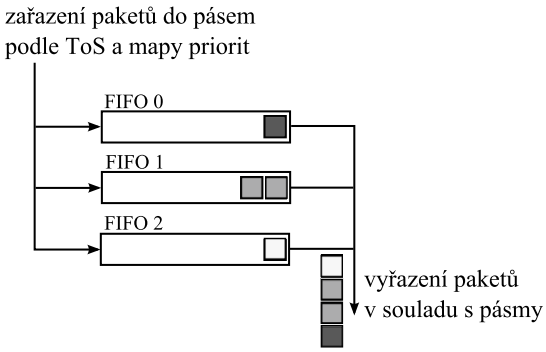
- FIFO,
- pfifo_fast,
- Token Bucket Filter,
- Stochastic Fair Queueing,
- Enhanced Stochastic Fair Queueing,
- Random Early Detection.

Každý z těchto algoritmů může být použit jako primární na rozhraní nebo uvnitř listové třídy classful qdisc. Z classless qdisc bude popsána pouze pfifo_fasf, jelikož se jedná o defaultní qdisc nasazovanou na rozhraní.

pfifo_fast

Funguje jako fronta FIFO (First In First Out) s možností priorizace dat. Rozděluje tok do tří pásem (FIFO front) podle nastavení bitů v ToS bytu každého IPv4 paketu. Princip je jednoduchý - pakety v pásmu 0 jsou vždy odeslány přednostně. Pakety v pásmu 1 jsou odeslány pouze v případě, že pásmo 0 je prázdné. Odeslání paketů v pásmu 2 je možné v případě prázdných pásem 0 a 1. Priorita tedy klesá s rostoucím číslem.

Mechanismus pfifo_fast je vhodný například pro poskytnutí vyšší priority přenosu hlasu s malými nároky na šířku pásma před ostatním provozem. Pokud by však byl priorizován tok dat s vyššími nároky na šířku pásma, mohl by zahltit celou linku a tak znemožnit ostatním tokům průchod síťovým prvkem.



Obr. 1.3: Struktura pfifo_fast qdisc

1.2.5 Classfull qdisc

Tyto disciplíny jsou používány pro tvarování a plánování různých druhů dat. Známými Linuxovými implementacemi jsou Prio, CBQ, HTB a HFSC. V porovnání s classles qdiscs, classful obsahují třídy, ze kterých se sestavuje celá hierarchie. Příklad takové klasické hierarchie s komponenty traffic controlu lze vidět na obr. 1.2.

Požadavek na vstup do fronty

Přesměrované, případně lokálně vytvořené pakety vstupují do root qdisc. Na identifikátory qdisců se připojují filtry, jimiž lze pakety přeposlat do konkrétních tříd a podtříd[7]. Pakety jsou po průchodu filtry, a tedy i celou hierarchií, umístěny do fronty listové třídy, kde čekají na odebrání. Každá listová třída defaultně disponuje FIFO frontou, kterou lze změnit za jakoukoliv jinou qdisc.

Požadavek na výběr z fronty

Tvarování může být uskutečněno classful qdisc jako je CBQ a HTB. Pokud jádro rozhodne o vyjmutí paketů na rozhraní, požadavek dostane pouze root qdisc, který posléze prohledává strom. Systém „vyhledávání“ vhodných paketů z front přímo k odeslání na rozhraní se může lišit v závislosti na typu classful qdisc. Vnořené třídy komunikují pouze s rodiči, nikdy přímo s rozhraním. Díky tomu třídám není umožněno vyřadit pakety z fronty vyšší rychlostí, než jakou povolí jejich rodiče. Tím můžeme provádět tvarování a případně plánování.

Disciplína **Prio** ve skutečnosti netvaruje, pouze plánuje. Hodí se zejména pro zahlcenou linku nebo zavedení do jiné classfull qdisc, která provádí tvarování.

CBQ *Class Based Queueing* je nejstarší a nejkomplexnější disciplínou umožňující tvarování. Princip je následující: pokud se snažíme tvarovat 10 Mbit/s linku na

[7] u HTB se doporučuje připojovat všechny filtry na root qdisc.

1 Mbit/s, měla by být nečinná 90 % času. Jestliže tomu tak není, musíme provoz zregulovat takovým způsobem, abychom dosáhli nečinnosti po dobu oněch 90 %. Vychází se z intervalu mezi odesláním dvou následujících paketů v rozlišení mikrosekund, patřící měřené třídě. Tento čas se porovná s očekávanou dobou, vypočítanou z požadované rychlosti a znaménko odchylky určuje, zda je třída pod nebo nad svým limitem. Kromě složitosti měření mezipaketové mezery v dostatečném rozlišení potřebuje CBQ znát přesnou rychlost fyzického rozhraní. U WLAN přenosová rychlost může značně kolísat v závislosti na mnoha podmínkách (odstup signál - šum, vzdálenost od klienta, aj.), což našim účelům nevyhovuje.

Praktická část práce se bude zabývat metodami HTB a HFSC. HTB eliminuje výše uvedené nedostatky CBQ. Navíc její konfigurace je transparentnější. HFSC umožňuje kromě stanovení přenosové rychlosti definovat také hodnoty zpoždění pro konkrétní provozy.

1.3 HTB Hierarchical Token Bucket

HTB je považována za srozumitelnější, intuitivnější a rychlejší náhradu CBQ. Obě metody jsou schopny použít jednu fyzickou linku k simulaci několika pomalejších a zasílat různý provoz do konkrétních simulovaných linek. Nejdříve je nutné určit, jak rozdělit fyzickou linku a jak rozhodnout, kterou simulovanou linku použít pro odeslání daného paketu. Cíle, které si klade HTB, jsou podmnožinou cílů CBQ, proto lze HTB chápat jako druh CBQ.

HTB odstraňuje nevýhody CBQ zavedením tzv. Leaky Bucketu LB. Každá třída obsahuje dva LB, z nichž první je definován parametry r [byte/s] a $burst$ [byte], druhý parametry c [byte/s] a $cburst$ [byte]. Pokud je aktuální hodnota toku $a(t)$ menší než hodnota garantovaného r či maximálního toku c, LB si uloží počet neodeslaných bytů do paměti, jejíž maximální velikost je dána právě parametrem $burst$, respektive $cburst$. V případě požadavků přenosu dat nad limit je LB schopen odeslat tolik bytů vyšší rychlostí, kolik si jich nastřádal do paměti. Hodnota parametru $burst$ se musí nacházet v mezích vztahu 1.1. Implicitně je nastavena na nejnižší možné číslo, lze ji však měnit.

$$burst \geq \Delta t \times r, \tag{1.1}$$

kde Δ je rozlišení systémového časovače.

Před popisem implementace výběru paketu si zavedeme doplňující terminologii:

- **třída** je uzlem stromu s parametry: garantovaný tok r rate, maximální tok c ceil, priorita $prio$, $quantum$ a úroveň v hierarchii,
- **plná listová třída** obsahuje pakety k odeslání,
- **plná vnitřní třída** má potomka typu plná listová třída,

- **podlimitní třída** má aktuální tok $a(t)$ menší než jeho r,
- **nespokojená třída** je současně plná a podlimitní, má nárok na obsloužení,
- **úroveň** určuje pozici třídy v hierarchii. Číslování začíná vzestupně od 0.

Stav třídy se popisuje barvami semaforu podle velikosti aktuálního toku $a(t)$:

- **červená** $a(t) > c$, třída přesáhla strop $\Rightarrow$ nemůže odeslat paket ani si půjčit od rodiče,
- **žlutá** $c \geq a(t), \geq r$, třída si může půjčit od rodiče,
- **zelená** $a(t) < r$, třída je pod limitem.

1.3.1 Výběr paketu a systém půjčování v hierarchii

Algoritmus pro výběr dalšího paketu k odeslání je uveden v [7] následovně:
Ze všech plných listů volme takový, který by si při odeslání paketu mohl půjčit tok od rodiče na nejnižší úrovni. Pokud je takových listů více, vybereme ten s nejvyšší prioritou. Pokud máme stále více listů, střídáme je pravidelně podle poměru jejich r[8]. Algoritmus splňuje očekávané požadavky:

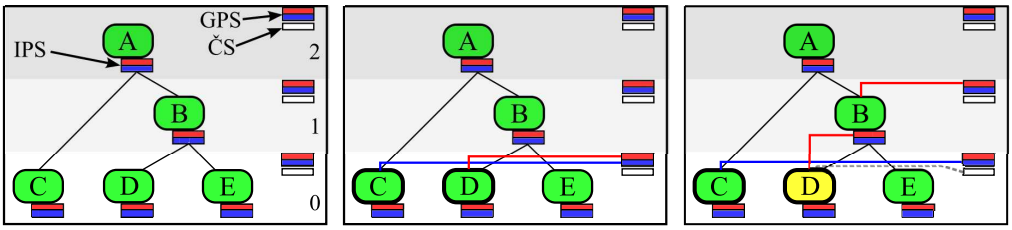
- toky tříd se stejnou prioritou si půjčují tok od společného rodiče v poměru jejich r,
- třída s vyšší prioritou má kratší odezvy a získá tok od společného rodiče přednostně,
- garantované toky jsou vždy splněny.

Pokud by byl výběr paketu uskutečněn procházením všemi třídami, dosáhli bychom lineární složitosti, což je nevyhovující. V paměti se proto ukládá aktuální stav a změny se provádějí pouze inkrementálně.

Každá vnitřní třída obsahuje interní půjčovací seznam IPS, který je rozdělen na tolik částí, kolik máme definovaných priorit (v našem případě 2). Jedna část vždy obsahuje informace o svých plných žlutých potomcích se stejnou prioritou. Dále se v každé úrovni nachází globální půjčovací seznam GPS a čekací seznam ČS. GPS udržuje informace o plných zelených třídách schopných okamžitě odeslat paket. Stejně jako IPS je GPS rozdělen na tolik částí, kolik existuje priorit. ČS obsahuje informace o červených a žlutých třídách, které momentálně nemohou odeslat paket a čekají na vypršení časovače. Po uplynutí této doby se změní jejich barva. Dle algoritmu probíhá výběr paketu následovně:

1. směrem od spodní úrovně se hledá neprázdný GPS,
2. po nalezení se vybere jeho část s nejvyšší prioritou,
3. postupuje se po trase, sestavené konkrétními IPS až k listové třídě.

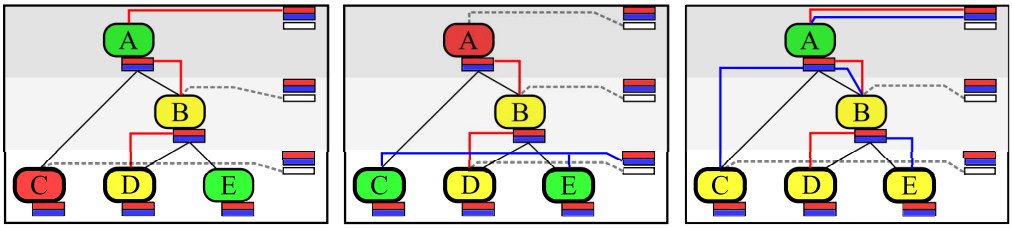
[8] list si může půjčit sám od sebe.



Obr. 1.4: Chování algoritmu při výběru paketu I

Příklad hierarchie HTB o třech úrovních s očíslováním 0, 1, 2 a šesti třídách znázorňuje obr. 1.4. Předpokládejme, že třída D má nejvyšší prioritu a plné listové třídy jsou značeny tučnými okraji.

Nejprve nejsou žádné pakety k odeslání (schéma vlevo). Na prostředním schématu přijdou pakety do listových tříd C a D. Protože jsou tyto třídy plné a zelené, napojí se podle priorit do GPS. Jestliže by nyní přišel požadavek k výběru paketu, algoritmus by podle výše uvedených kroků rychle zjistil, že k odeslání bude vybrána třída D; případně C po odčerpání všech paketů z D. Po odeslání paketu z D může dojít k překročení limitu r (situace vpravo). V tomto případě se D připojí do ČS a IPS vyšší priority u B. Vnitřní třída B je napojena na GPS, což by umožnilo D zapůjčením od B vysílat pakety. Výběr paketu vždy začíná nejnižší úrovní prohledáváním GPS, kde se nyní nachází plná zelená třída C a proto je zvolena, i když má menší prioritu než D. Vyšší úrovně se již neprohledávají. C má tedy nárok na svůj garantovaný tok a není důvod upřednostňovat D.



Obr. 1.5: Chování algoritmu při výběru paketu II

Obr. 1.5 vlevo znázorňuje případ, kdy se třída C dostane na ČS po překročení stropu c odebráním paketů a třídu B, která se po změně na žlutou napojí na IPS třídy A. Třída A je připojena na GPS své úrovně a červené třídě C není umožněno ani posílat pakety, ani si půjčovat od rodiče. Právo odesílat pakety má pouze D. Situace uprostřed ukazuje, že cesta mezi IPS existuje i v případě, kdy nejvyšší třída nemá s GPS spojení. C a E jsou napojeny na stejnou část GPS a odebírání probíhá v poměru jejich r. Spravedlivé střídání odebírání je zajištěno ukládáním identifikátorů tříd - classid do seznamů. Tyto seznamy jsou implementovány pomocí Red-Black stromu. Na schématu vpravo je demonstrována možnost připojení

vnitřních tříd na stejný IPS rodiče pro více priorit, zatímco listové třídy mohou mít pouze jedno připojení. Vzhledem k vyšší prioritě je pro výběr paketu zvolena třída D. Jestliže by však byla priorita všech listových tříd stejná, nastalo by spravedlivé rozdělování. [7, 8]

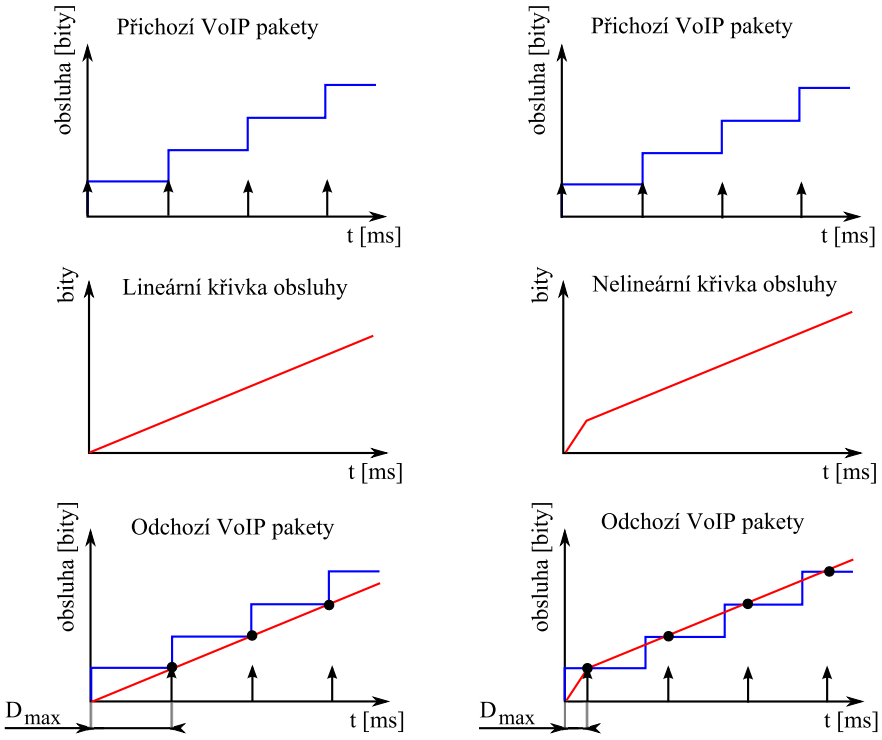
1.3.2 Použití tc u HTB

Následující výčet rekapituluje parametry tc pro použití u HTB:

- `default` - označení třídy pro zasílání neklasifikovaných paketů minor identifikátorem, volitelný parametr u každé qdisc,
- `r2q` - hodnota pro případný výpočet parametru quantum uživatelem,
- `rate` - garantovaná přenosová rychlost r přidělená třídě, třída si po překročení r může stále půjčovat,
- `burst` - maximální počet bytů ve shluku, které se nashromažďuji během intervalu s $a(t) < r$,
- `ceil` - maximální přenosová rychlost c, tzv. strop. Třída si již nemůže půjčovat po překročení c,
- `cburst` - maximální počet bytů ve shluku, které se nashromažďuji během intervalu s $a(t) < c$,
- `mtu` - maximální velikost odesílaného paketu,
- `prio` - priorita listové třídy; nižší hodnoty jsou obsluhovány přednostně,
- `quantum` - počet bytů obsloužených z listové třídy najednou. Výpočet provádí HTB, ale lze nastavit manuálně. Vychází se ze vztahu `quantum`$= (\frac{\texttt{rate}}{\texttt{r2q}})$.

1.3.3 Channel aware HTB

Většina komerčních 802.11 zařízení do svých struktur zavádí jednoduchou frontu fifo, nedovolující rozdělovat toky pro různé příjemce. Pokud vzdálenější stanice s horšími parametry radiového kanálu přenáší data, automatické přizpůsobení přenosové rychlosti spolu s vícenásobnými pokusy o zahájení spojení mohou v důsledku způsobit ve směru downlink podstatné snížení propustnosti uvnitř AP, což zhoršuje výkonnost celého systému. Stanice obsadí celý kanál po delší dobu a tím zabrání k dočasnému přístupu ostatním žadatelům. Pro překonání těchto nechtěných vlastností systému 802.11 a dosažení sdílené šířky pásma, která je rozdělena v pevném poměru ve směru download mezi mobilními stanicemi využívající jeden přístupový bod s Linuxem, byly experimentálně navrženy a implementovány plánovací algoritmy WHTB i TWHTB. Oba vychází z principu HTB s rozšířením o možnost brát v úvahu informaci o kvalitě rádiového kanálu jednotlivých stanic a transportní třídě služeb



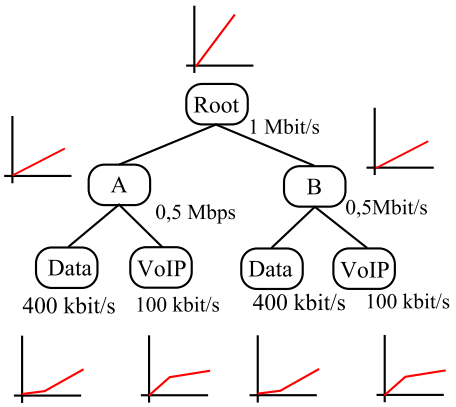
Obr. 1.6: Funkce lineární a nelineární křivky obsluhy

požadovanou koncovým uživatelem. TWHTB se od WHTB odlišuje v postupu odhadu stavu kanálu – namísto měření SNR analyzuje čas mezi úspěšným zasláním rámce včetně případných opakování a příjmu zprávy ACK. [9]

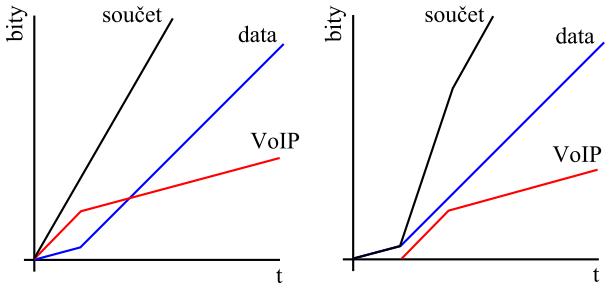
1.4 HFSC - Hierarchical Fair Service Curve

Plánovací algoritmus HFSC umožňuje kromě rozdělovaní šířky pásma linky mezi uživatele či služby také odděleně řídit maximální dobu zpoždění koncovým stanicím. Toho lze využít v případě, kdy se v síti vyskytují přenosy dat citlivých na zpoždění spolu s dalším intenzivním provozem. Metoda HFSC se snaží maximálně aproximovat teoretický model Fair Service Curve - Link Sharing, jehož úkolem je současná záruka křivek obsluhy všech uzlů a distribuce nadbytečné šířky pásma nevyužité třídou spravedlivě mezi ostatní třídy.

Křivku obsluhy *service curve* lze definovat jako minimální množství obsluhy, které by mělo být poskytnuto třídě. Zavedením nelineární křivky obsluhy dosáhneme v porovnání s lineární zrušení vazby u přidělování šířky pásma a zpoždění. Nejčastěji se zavádějí po částech lineární neklesající funkce. Obecně bude mít konkávní křivka za výsledek nižší průměrné a nejnižší možné zpoždění oproti lineární nebo konvexní křivce.



Obr. 1.7: Hierarchie tříd HFSC



Obr. 1.8: Krátkodobé nedodržení křivky obsluhy

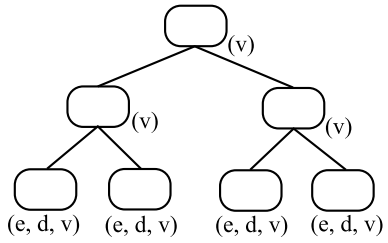
Obr. 1.6 znázorňuje, jak lze dosáhnout zkrácení maximální čekací doby D_{max} pro odeslání VoIP paketů zavedením dvousložkové konkávní křivky obsluhy. Doba D_{max} v podstatě představuje maximální možný časový interval mezi příjmem a odesláním konkrétního paketu.

Příklad hierarchie HFSC je na obr. 1.7. Celková kapacita linky je rozdělena mezi uživatele A a B a dále mezi jejich služby. Každá z tříd má vlastní křivku obsluhy. Je zřejmé, že snížené zpoždění VoIP provozu uživatele B bude mít za důsledek zvýšení zpoždění u jeho datové služby.

Jak již bylo uvedeno, FSCLS je ideální model, který však nelze realizovat se stoprocentní úspěšností. Metoda HFSC jej aproximuje spolu s definicí okolností, za jakých není možné současně splnit všechny požadavky. Obr. 1.8 vpravo představuje takovou situaci. Třída vysílá data nenáchylná na dobu zpoždění s maximální přenosovou rychlostí 400 kbit/s. Pokud VoIP třída se 100 kbit/s provozem začne přenášet data později než jeho sourozenec, dojde ke krátkodobému překročení křivky obsluhy rodičovské třídy.

1.4.1 Algoritmus pro výběr paketu

Plánování je založeno na kritériích *real-time*, zajišťující garanci obsluhy všech listových tříd a *link-sharing*, které usiluje o splnění všech křivek obsluhy vnitřních tříd a o spravedlivou distribuci nadměrné šířky pásma. *Real-time* kritérium bude použito pro výběr paketů pouze v případě potenciálního rizika narušení garance služby listových tříd. V jiném případě bude zvoleno *link-sharing* kritérium. Takový přístup zajistí real-time požadavky listových tříd současně s minimalizací vzniku nesouladu mezi aktuální dosaženou obsluhou vnitřních tříd a obsluhou, definovanou v ideálním modelu.



Obr. 1.9: Výskyt parametrů

Všechny listové třídy udržují trojici časových parametrů (e, d, v) zatímco vnitřní třídy pouze parametr v. Důvodem je, že *deadline* d a čas oprávnění *eligibility time* e slouží pro garanci křivek obsluhy, přičemž HFSC poskytuje garanci právě listovým třídám. Na druhou stranu virtuální čas v je používán pro hierarchické sdílení linky v celé struktuře, a proto jej udržují všechny třídy.

Prvnímu paketu v každé frontě listové třídy čekajícímu na odeslání náleží hodnoty e a d. Pokud jsou d paketů listové třídy dodrženy, její křivka obsluhy je zajištěna. Pomocí času oprávnění e se rozhoduje mezi výše uvedenými kritérii pro výběr dalšího paketu. Paket, čekající na odeslání je považován za oprávněný, pokud je $e \leq t$, kde t je aktuální čas. V případě, že se ve struktuře tříd vyskytuje oprávněný paket, existuje riziko konfliktu mezi kritérii, přičemž z důvodu vyšší priority je vybráno *real-time* kritérium. Následně dojde k procházení listových tříd pro odeslání paketu s nejmenším d. Jestliže ve struktuře není žádný oprávněný paket, za použití *link-sharing* podmínky se prohledávají třídy směrem od kořenu až k listu. Poté je odeslán nalezený paket s nejmenší hodnotou v, která představuje normalizované množství služby, přijaté třídou. V ideálním spravedlivém systému by měly být všechny virtuální časy sousedů stejné[9]. V implementaci jsou parametry e, d a v počítány pomocí křivek E(), D()a V(). Aktualizace křivek je nutná pouze při přechodu relace z pasivního do aktivního stavu. Popis algoritmu pro výběr paketu k odeslání shrnuje následující pseudokód:

```
if (existuje oprávněný paket)
   /*real-time kritérium*/
   pošli vhodný paket s minimální čekací dobou d;
else
   /*link-sharing kritérium*/
    pošli paket s minimálním virtuálním časem v;
```

Nejhorší možná složitost algoritmu $O(\log n)$ je podobná ostatním plánovačů paketů. [12, 13]

[9]časy d a e jsou reálné časy vztažené ke skutečným hodinám, zatímco v spíše představuje relativní čas vzhledem k množství poskytnuté služby rodičem.

1.4.2 Příklad použití HFSC na Linuxu

Pro vytvoření metody je nejdříve nutné přiřadit HFSC qdisc na síťové rozhraní spolu s volitelnou specifikací výchozí třídy dle příkazu:

```
tc qdisc add dev DEV root handle ID: hfsc [default CLASSID]
```

Následuje definice hierarchie tříd se specifičtějšími parametry:

```
tc add class dev DEV parent PARENTID classid ID hfsc [[ rt SC ]
[ls SC] [sc SC]]
```

Konfigurací křivek obsluhy lze určit konkrétní vlastnosti každé z tříd:

```
SC := [umax bytes dmax ms] rate BPS
```

Listovým třídám je možno přiřadit křivky real-time `rt` a link-sharing `ls`, zatímco vnitřním třídám pouze link-sharing. Obě křivky lze současně definovat pomocí parametru `sc`. Zavedením `ul` za poslední uvedený příkaz spolu s číselnou hodnotou určíme maximální poskytnutou přenosovou rychlost - podobně jako s `ceil` u HTB. Křivka obsluhy je popsána rychlostí jejího vysílání, čili jejím sklonem. Specifikaci křivky složené ze dvou částí se provádí parametrem maximálního zpoždění `dmax` při odeslaných `umax` bytů. [13]

1.5 Doplněk IEEE 802.11e

Technologie Wi-Fi standardu IEEE 802.11a, b, g byly při svém vzniku primárně určeny k poskytovaní klasických datových služeb *best effort*. Nezahrnují v sobě podporu tříd služeb, podporu vyjednávání QoS podmínek koncových stanic s přístupovým bodem, ani podporu pro řízení přístupu. Standardy specifikují dvě přístupové metody k médiu na MAC podvrstvě linkové vrstvy. První a zároveň povinnou je distribuovaná koordinační funkce DCF, druhou centralizovaná koordinační funkce PCF. U DCF jednotlivé mobilní stanice soutěží o přístup, u PCF připouští k médiu stanice přístupový bod. U kombinovaného DCF/PCF módu odesílá přístupový bod v definovaném intervalu rámec *beacon*, dále následuje interval se soutěžením s použitím metody DCF a poté interval bez soutěžení - metoda PCF.

Bezdrátové lokální sítě WLAN se postupem času rozšířily do firem a domácností, přičemž vzrůstající nároky uživatelů na přenos multimediálních datových toků přes WLAN vyústily ve vznik doplňku IEEE 802.11e, který eliminuje výše uvedené nedostatky vylepšením MAC podvrstvy. IEEE 802.11e zavádí přístupovou techniku, zvanou hybridní koordinační funkce HCF, která nahrazuje DCF. HCF se dále dělí na:

- EDCA - rozšířený distribuovaný přístup ke kanálu,
- HCCA - přístup ke kanálu řízený HCF.

1.5.1 EDCA

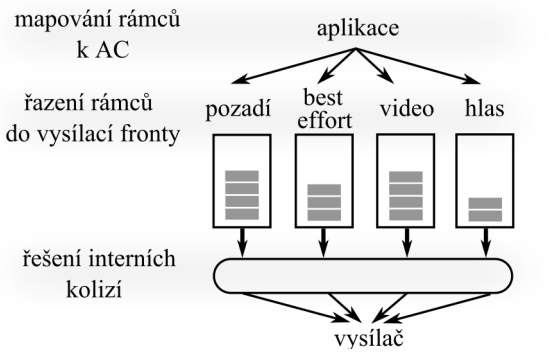
Tato technika je v podstatě rozšířením DCF. Vylepšení spočívá v prioritním plánování paketů v závislosti na prioritách provozu standardu 802.1D, které jsou provázány se čtyřmi kategoriemi přístupu AC v EDCA, viz tab. 1.3. Jedné kategorii přístupu tedy může současně odpovídat i více prioritních hodnot. Pakety jsou klasifikovány aplikací do kategorií dle značky DSCP nebo 802.1D[10]. Každá ze čtyř přístupových kategorií má přiděleny vlastní hodnoty CW_{min}, CW_{max}, AIFS, TXOP. CW_{min} a CW_{max} udávají minimální, respektive maximální velikost okna soutěžení. AIFS je časový interval mezi uvolněním média a začátkem soutěžení o přístup mezi stanicemi. TXOP specifikuje maximální dobu, po kterou stanice mohou vysílat rámec.

Tab. 1.3: Souvislost priorit 802.1D s kategoriemi přístupu v EDCA

IEEE 802.1D		EDCA	
Priorita	Návrh použití	Kategorie přístupu	Návrh použití
1 min	pozadí	0	pozadí
2	rezerva	0	pozadí
0	best-effort	0	best-effort
3	excellent-effort	1	best-effort
4	řízená zátěž	2	video
5	video	2	video
6	hlas	3	hlas
7 max	řízení sítě	3	hlas

V praxi na vysílací stanici paket po označení určitým DSCP aplikací vstupuje do jedné ze čtyř nezávislých front, kde každá fronta odpovídá jedné AC, viz obr. 1.10. Klient řeší vnitřní kolize mezi různými frontami a poté se princip opakuje již mezi vysílacími stanicemi.

[10]IEEE 802.1D je standard pro pevné lokální sítě.



Obr. 1.10: Vysílací fronty uvnitř WME stanice

1.5.2 HCCA

Metoda vychází z principu PCF, navíc může vysílat i v intervalu se soutěžením. Stanice musí nejdříve definovat svoje požadavky na datový tok ve specifikaci provozu TSPEC, které jsou zaslány přístupovému bodu. Řízení přístupu v AP poté rozhodne o zamítnutí či povolení požadavků v TSPEC. V porovnání s technikou EDCA, rozhodující se na základě relativních priorit, má HCCA vyšší prioritu a dokáže zcela garantovat dobu přenosu nebo zpoždění.

1.5.3 Tvarování provozu a WMM

WME *Wireless Multimedia Extensions* je uváděna také pod názvem WMM *Wi-Fi MultiMedia* a funkcí odpovídá přístupové metodě EDCA z doplňku 802.11e. Účelem této certifikace vydanou Wi-Fi aliancí bylo poskytnutí podpory QoS v bezdrátových sítích před schválením doplňku 802.11e. Dále se práce bude zaměřovat na funkci WME, jelikož v současnosti jsou na trhu dostupné bezdrátové karty podporující certifikaci WME, ne však další funkce z 802.11e.

S odkazem na obr. 1.10 předpokládejme následující situaci. V jedné frontě např. na přístupovém bodu čeká 50 paketů od jedné stanice a přichází 1 paket od stanice další. Tento paket musí čekat celou dobu před úspěšným odesláním všech padesáti paketů, stojících ve frontě před ním. V případě zahlcení fronty také může dojít k jeho zahození. Proto nelze pomocí WME spravedlivě řešit rozdělení pásma pro různé stanice.

WME je způsob, který dovoluje snížit zpoždění pro určité pakety oproti klasickým standardům 802.11a,b,g, avšak nemůže vyřešit problém zpoždění vnitřních front. Tvarování provozu mezi klienty a prioritní systém WME jsou tedy dvě odlišné záležitosti, které však spolu nikterak nekolidují a mohou se vzájemně doplnit.

Algoritmy pro tvarování provozu jako jsou HTB a HFSC potřebují ke správné funkci znát maximální přenosovou rychlost linky, na jehož rozhraní pracují. Metody byly navrženy pro Ethernetové spoje, problém ale vzniká u bezdrátových sítí, kde není možné předem přesně specifikovat maximální přenosovou rychlost, a to především z těchto důvodů:

- bezdrátové sítě jsou poloduplexní,
- přenosová rychlost značně kolísá v závislosti na rušení a vzdálenosti od AP,
- přenos rámce může být při neúspěchu několikrát opakován,
- přenos určitého objemu dat trvá u menších rámců delší dobu,
- RTS/CTS mechanismus dále snižuje přenosovou rychlost.

Pokud by byla nastavena vyšší rychlost než reálně dosažitelná, algoritmus by nepracoval správně. U Wi-Fi sítí se tedy nabízí dvě možnosti. Nastavit hodnotu tvarované přenosové rychlosti přehnaně optimisticky - v tom případě bude algoritmus fungovat jen občas, nebo hodnotu poddimenzovat s tím, že přicházíme o část kapacity. Projekt NFX, zabývající se úpravou WiFi ovladačů čipsetů Atheros, se snaží mj. implementovat tvarování na MAC podvrstvě linkové vrstvy, což by tuto neutěšenou situaci mohlo v blízké budoucnosti vyřešit. [2, 11, 14, 15]

2 SIMULACE DOPLŇKU 802.11E

Cílem simulace je ověření zajištění QoS u dat citlivých na zpoždění v bezdrátové síti s podporou doplňku IEEE 802.11e. Pro provedení je zvolena současná verze NCTUns 6.0, běžící na virtualizované linuxové distribuci Fedora 12.

2.1 NCTUns

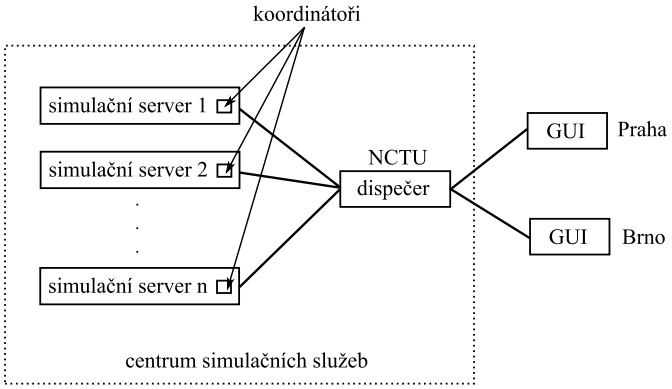
Jedná se o freewarový linuxový síťový simulátor a emulátor, který je schopen simulovat řadu protokolů s kvalitními výsledky pomocí přímého využívání skutečného protokolového TCP/IP zásobníku. Na prvcích v simulované síti mohou být uváděny do chodu reálné aplikace bez dodatečných úprav, dokonce i ve stádiu vývoje. Díky tomu lze generovat věrohodnější výsledky než v případě jednoduchých generátorů provozu a testovat vyvíjené aplikace před oficiálním uvedením na různé síťové podmínky. V tomto ohledu NCTUns překonává simulátory jako jsou NS2 nebo OPNET Modeler. Realizaci paketové komunikace mezi simulovanou a fyzickou sítí lze uskutečnit v módu emulace, který dovoluje zkoumat funkce i výkonnost reálných prvků. Nadměrnému vytížení výpočetních prostředků při práci v reálném čase v módu emulace je možno předejít rozprostřením zátěže do několika stanic. Provoz a konfigurace u simulované a reálné sítě se provádí téměř stejným způsobem. Tato provázanost umožňuje s výhodou NCTUns uplatnit pro výukové účely. Na simulované síti je také možno spustit skutečné linuxové nástroje pro konfiguraci a monitorování provozu. V souvislosti s QoS podporuje NCTUns např. simulaci DiffServ domény nebo bezdrátové sítě s podporou 802.11e. [10]

2.1.1 Architektura a komponenty

NCTUns může fungovat ve dvou módech, a to ve výchozím *single machine* nebo v módu *multi machine*. V prvním případě jsou všechny komponenty instalovány a spouštěny na jedné stanici. Ve druhém se výpočty rozdělují do farem simulačních serverů - viz obr. 2.1. Distribuovaná architektura NCTUns se skládá z osmi základních komponentů:

1. **GUI**,
2. **simulační stroj** - program poskytující simulační služby; stanice, na které běží, se označuje pojmem simulační server,
3. **protokolové moduly** - třídy v jazyku C++ pro podporu protokolů nebo funkcí jako je plánování paketů; moduly jsou vázány na simulační stroj,

4. **dispečer** - program, který komunikuje s GUI a koordinátorem, dokáže současně řídit a používat několik simulačních serverů pro zvýšení výkonnosti výpočtu simulace,
5. **koordinátor** - povinně spuštěný program na každém běžícím simulačním stroji, komunikuje s dispečerem o stavu simulačního stroje a práci na něm prováděné[1],
6. **démoni** - jsou uváděni do chodu po celou dobu simulace,
7. **reálné uživatelské programy**,
8. **záplaty jádra**.



Obr. 2.1: Architektura s komponenty NCTUns v *multi machine* módu

Vzdálený uživatel může na GUI předložit simulaci pro zpracování dispečeru, který ji přepošle do dostupného simulačního serveru[2]. Po dokončení zpracování server zašle zpět výsledek uživateli. [10]

2.1.2 Módy GUI

Pro ověření zajištění kvality služeb v síti 802.11e byl zvolen *single machine* mód. Rozběhnutí GUI předchází nastavení proměnných prostředí s následným spuštěním dispečeru a koordinátoru s právy roota. V GUI se lze pohybovat mezi čtyřmi následujícími pracovními módy:

Draw Topology

Slouží k rozmístění prvků a propojení mezi nimi. U bezdrátové sítě je nutné vybrat zařízení, patřící do stejné podsítě.

[1] výměna informací mezi GUI a simulačním strojem vždy prochází přes koordinátor.
[2] simulační server obsahuje simulační zdroj, koordinátor protokolové moduly a záplaty jádra.

Edit Property

Tento mód umožňuje definovat komunikující aplikace, protokoly, vlastnosti konkrétních prvků nebo výpadky linek. Protokolové moduly, jejich nastavení a výběr ukládaných hodnot pro vynesení do grafů lze upravit v *Node editoru*.

Po sestavení nebo změně topologie a přechodu do *Edit Property* módu se vždy znovu automaticky přiřadí všem rozhraním IP a MAC adresy. Jestliže již máme u objektů nastavenou komunikaci včetně adres, je dobré zkontrolovat, zda jednotlivá rozhraní odpovídají aktuálnímu přiřazení.

Run Simulation

Průběhu simulace by měla předcházet kontrola umístění dispečeru. U *single machine* módu se nastavuje jeho IP adresa v **G_settings→Dispatcher** na lokální smyčku 127.0.0.1. Jméno a heslo musí odpovídat právě přihlášenému uživateli. Výpočet simulace spustíme pomocí **Simulation→Run**. Simulační server po vykonání výpočtu předá výsledek v podobě několika souborů zpět do GUI.

Play Back

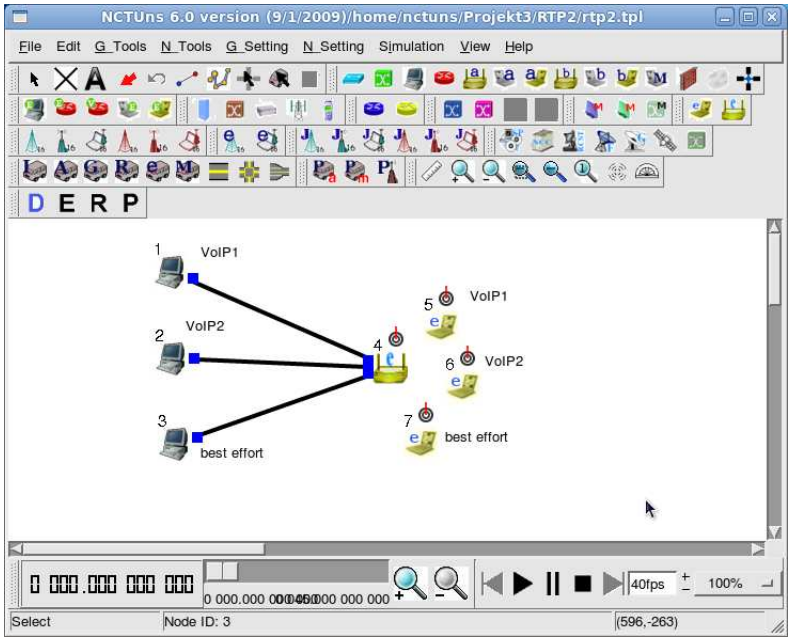
Mód slouží pro pozorování tras paketů. Vygenerované hodnoty pro grafy jsou uloženy do textových souborů. Vestavěný nástroj NCTUns pro vynášení grafů má pouze omezené možnosti - nedovoluje například upravovat výchozí velikost okna pro vykreslení a přidávat legendu delší než několik málo znaků.

2.1.3 Simulace EDCA

Struktura simulované bezdrátové sítě IEEE 802.11e s využitím rozšířeného distribuovaného přístupu ke kanálu EDCA je znázorněna na obr. 2.2. Sestává ze tří mobilních stanic a jednoho přístupového bodu.

Hlasová data jsou přenášena obousměrně mezi stanicemi $5-1$ a $6-2$ v časovém intervalu $10-40\,\mathsf{s}$, respektive $20-40\,\mathsf{s}$. Pro komunikaci mezi koncovými uzly jsou použity protokoly RTP, RTCP, SDP a kodek G.722. Vzorkovací kmitočet má u každého spojení hodnotu $8\,\mathsf{kHz}$, 8 bitů na vzorek a interval vysílání paketu $20\,\mathsf{ms}$. Příjem spolu s odesíláním RTP a RTCP paketů zajišťuje vestavěná aplikace NCTUns s názvem *rtpsendrecv*. Jelikož se jedná o přenos hlasu, bezdrátovým stanicím byla přiřazena prioritní úroveň 6.

Mobilní stanice 7 zahlcuje po celou dobu simulace přístupový bod objemným provozem směrem k uzlu 3. Priorita provozu je nastavena na úroveň 0, jedná se tedy o best effort.



Obr. 2.2: Topologie sítě 802.11e - EDCA

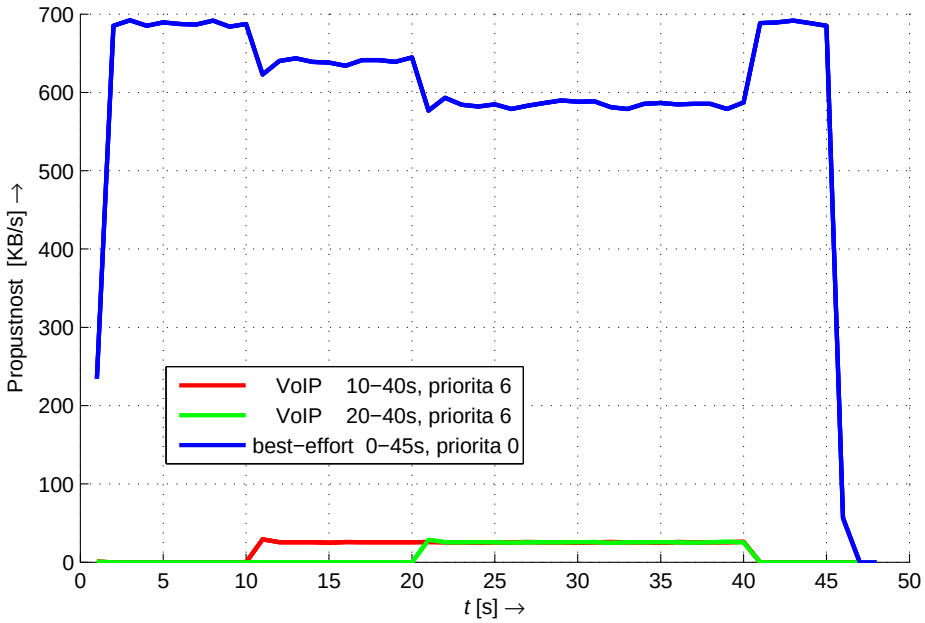
Výstup simulace představuje obr. 2.3. Jednotlivé průběhy znázorňují propustnost příchozích a odchozích paketů na konkrétních bezdrátových stanicích. Z grafu je zřejmá priorizace přenosu hlasu před best effort provozem. Zahájení prvního hovoru v čase 10 s a druhého v čase 20 s sníží propustnost stanice vysílající best effort.

2.1.4 Simulace koexistence EDCA s HCCA

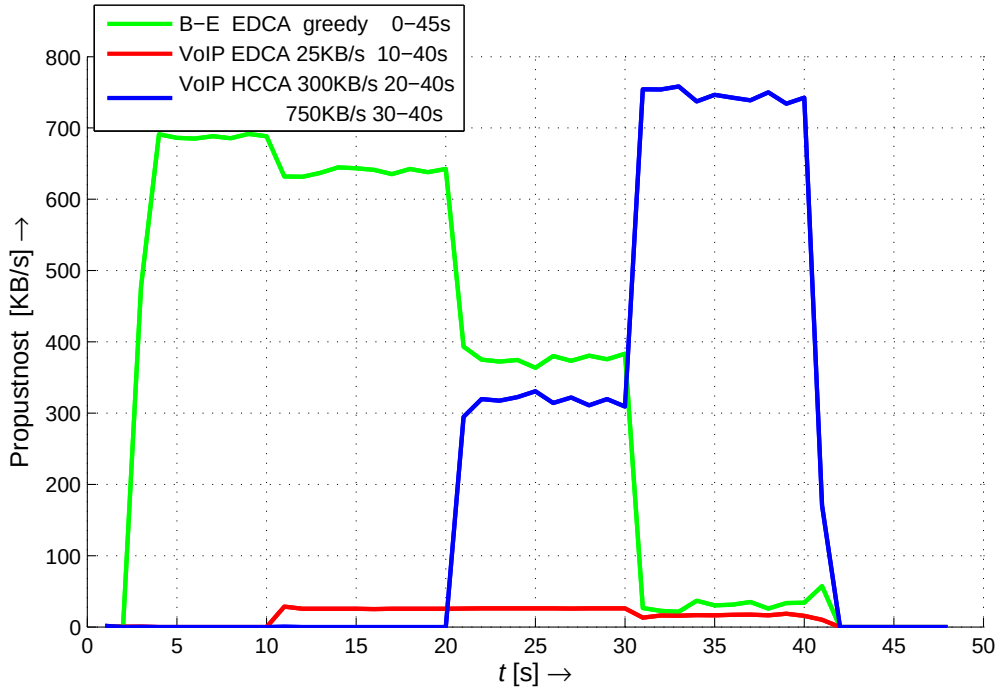
Tato simulace prověřuje vliv koexistence přístupových metod HCCA a EDCA. V topologii je nyní pět bezdrátových stanic. Dvě z nich pracují s EDCA, kdy první zasílá s prioritou 0 (best effort) tzv. greedy provoz - má tedy snahu zahltit objemnými daty přenosové médium. Druhá má prioritu 6 (hlas) s přenosovou rychlostí cca 25 KB/s.

Zbývající bezdrátové stanice přenášejí hlas s využitím metody HCCA. Všechny nejdříve musí definovat svoje požadavky na datový tok ve specifikaci provozu TSPEC, které jsou zaslány přístupovému bodu. V NCTUns má každá z těchto stanic v souvislosti se specifikací provozu definován maximální servisní interval 20 ms a nominální velikost paketu 1024 B. Přenosové rychlosti v řádu stovek KB/s mají představovat hned několik navázaných spojení tak, aby bylo lépe vidět vyšší zatížení sítě.

Výsledek simulace je znázorněn na obr. 2.4. Best effort provozu je postupně snižována propustnost zahajováním přenosů hlasu dalších stanic. V čase 20 s je vidět, jak VoIP s HCCA sníží propustnost best effort provozu s EDCA, ale VoIP s EDCA vzhledem k vyšší relativní prioritě zůstává na stejné hodnotě. V čase 30 s je spuštěn další VoIP provoz s HCCA. Z průběhu grafu je také zřejmé, jak v tomto okamžiku



Obr. 2.3: Propustnost na stanicích sítě IEEE 802.11e s metodou EDCA



Obr. 2.4: Propustnost na stanicích sítě IEEE 802.11e, HCCA a EDCA

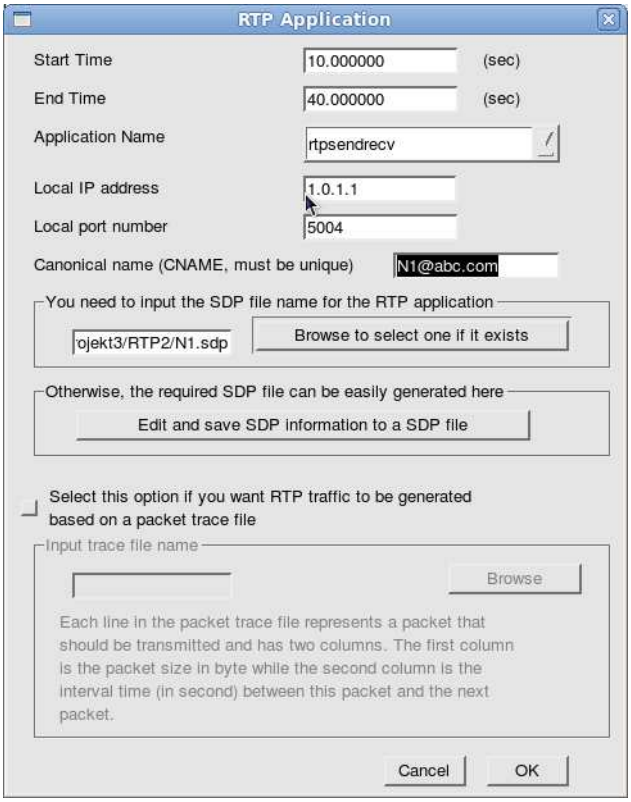
VoIP s HCCA sníží propustnost oběma přenosům - VoIP i best effort s EDCA.

Závěrem lze tedy říci, že při nadměrném zatížení bezdrátové sítě 802.11e s uplatněním přístupových metod EDCA a HCCA pro přenos hlasu může dojít k situaci, kdy provoz s HCCA potlačí provoz s EDCA. Pro přenos hlasu či videa je tedy vhodnější volit HCCA s definicí specifikace provozu, než-li EDCA s odpovídající prioritou.

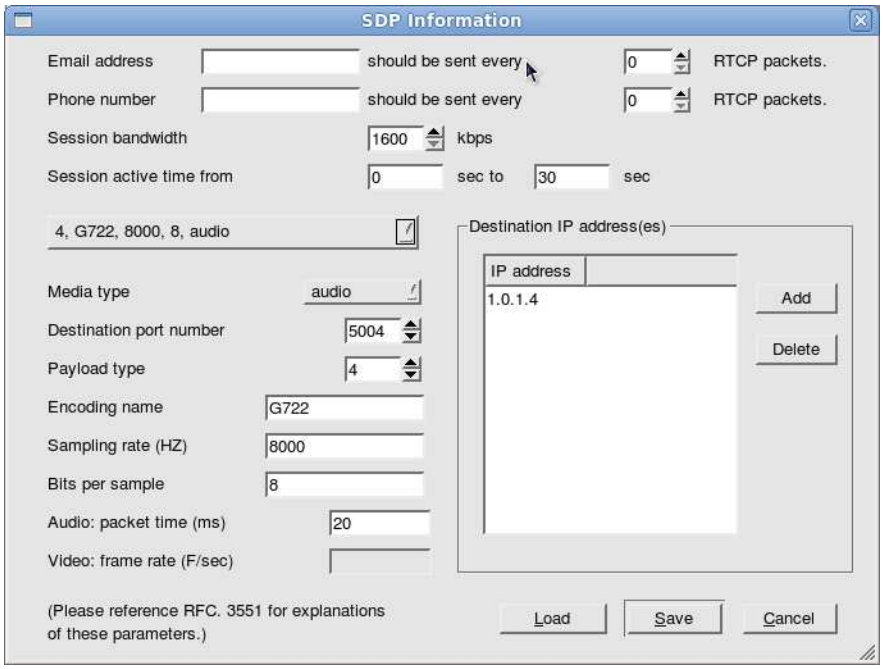
2.1.5 Poznámky k nastavení

Dialogové okno pro nastavení RTP aplikace je na obr. 2.5. Vyvoláme jej dvojklikem na konkrétní uzel a poté zvolíme „Add RTP“. Zde se nastavuje čas spuštění a zastavení aplikace, jméno aplikace, IP adresa s volným portem aktuálně konfigurovaného uzlu a kanonické jméno. Dále by měl být vytvořen SDP konfigurační soubor sestávající z informací o sezení a médiu, který bude čten RTP aplikací. Jeho vytvoření lze dosáhnout v dialogovém oknu RTP aplikace kliknutím na položku „Edit and save SDP Information to a SDP file“, čímž se otevře okno na obr. 2.6. Zde je mimo jiné možno nastavit aktivní interval sezení v rámci dříve stanoveného intervalu RTP aplikace, typ kodeku, přenášené informace, cílový port, vzorkovací kmitočet, počet bitů na vzorek a interval odesílání audio paketů. Důležitým je zejména seznam IP adres cílových stanic. Na těchto stanicích je také potřeba nastavit příslušnou RTP aplikaci.

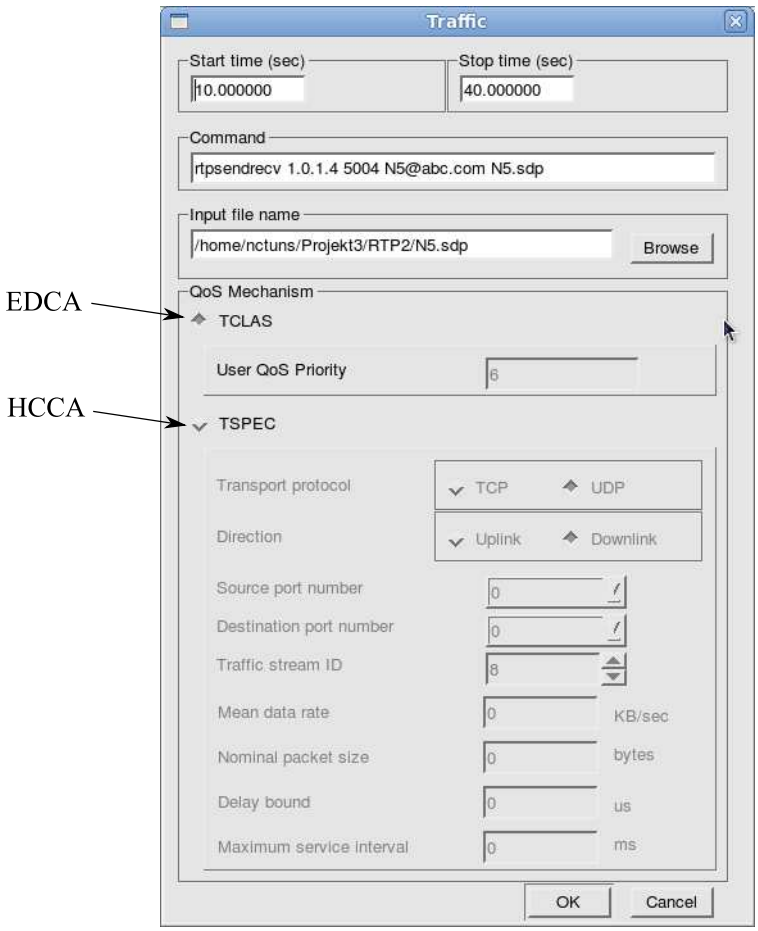
Volba mezi EDCA a HCCA se provádí v modifikačním okně aplikace stejně tak jako jejich nastavení - viz obr.2.7. V případě EDCA se volí QoS priorita třídy provozu. U HCCA se nastavují parametry vyjednávané s přístupovým bodem.



Obr. 2.5: Dialogové okno RTP aplikace



Obr. 2.6: Dialogové okno SDP



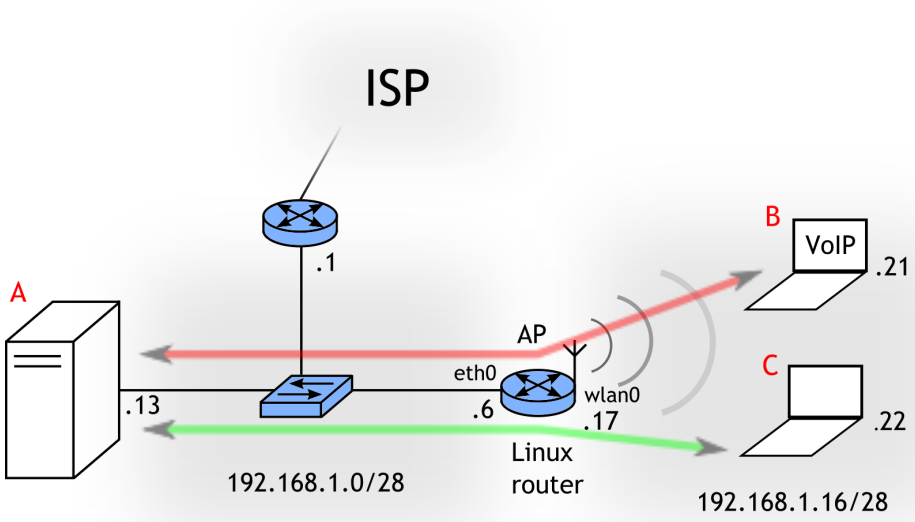
Obr. 2.7: Volba mezi EDCA a HCCA

3 PRAKTICKÁ REALIZACE

Praktická realizace je rozdělena do čtyř částí. První popisuje použité měřicí pracoviště, druhá se zabývá zajištěním QoS v reálné bezdrátové síti v souvislosti s doplňkem 802.11e, respektive se specifikací WME. Zbývající dvě části jsou věnovány metodám pro tvarování provozu pomocí HTB a HFSC.

3.1 Měřicí pracoviště

Struktura pracoviště je zobrazena na obr. 3.1. Šipky naznačují trasy měřeného síťového provozu a jejich barvy korespondují s naměřenými průběhy ve výsledných grafech. Na všech stanicích je nainstalována linuxová distribuce Debian Squeeze - jádro 2.6.32. U bezdrátových stanic B a C, z výroby vybavených kartami s podporou WME, se jedná o virtualizaci v programu Virtualbox. Jejich vzdálenost od AP (Linux router) činí 4 m.



Obr. 3.1: Uspořádání měřicího pracoviště

Z důvodu vysokých cen komerčního softwaru pro měření síťového provozu byl výběr omezen na volně dostupné programy Mausezahn, Iperf a D-ITG. Jako ideální se jevil program Mausezahn, kterým je mj. možné simulovat VoIP provoz, tvořit statistiky a pracovat v interaktivním módu s ovládáním velmi podobnému Cisco IOS, avšak při pokusných měřeních se ukázal jako nestabilní. Stejně jako programu Iperf, chybí také Mausezahnu možnost měření jednosměrného zpoždění. Proto byl vybrán

program D-ITG, který již s jednosměrným zpožděním dokáže pracovat, je však třeba zajistit synchronizaci mezi vysílací a přijímací stanicí. Stavu synchronizace lze dosáhnout pomocí GPS přijímačů nebo protokolem NTP. Zvolena byla druhá možnost - instalace NTPd programu na A a B s tím, že došlo k patřičné úpravě jejich konfiguračních souborů. Počítač A byl nastaven jako NTP klient pro aktualizaci času ze serverů na internetu a současně jako NTP server pro synchronizaci času s bezdrátovou stanicí B v režimu NTP klient. Na stanici B bylo dále nutné zablokovat časovou synchronizaci s hostitelským systémem Windows nástrojem VBoxManage. Na příkazové řádce OS Windows v instalačním adresáři toto zajistil příkaz:

```
vboxmanage setextradata [jméno VM] \
  "VBoxInternal/Devices/VMMDev/0/Config/GetHostTimeDisabled" "1"
```

Před každým měřením byl vypsán status démona na B (VoIP) klientovi příkazem `ntpq -pn` spolu s poznamenáním korekční hodnoty `offset`, viz:

```
     remote           refid      st t when poll reach   delay   offset  jitter
==============================================================================
*192.168.1.13 213.151.89.43    3   u   38   64  377    1.934    0.802   0.447
```

Na výše uvedeném příkladu značí hvězdička referenční časový zdroj, položka `remote` udává jeho IP adresu, `when` dobu v sekundách od posledního dotazu, `poll` interval mezi dotazy a `reach` s hodnotou 377 informuje o posledních deseti úspěšných odpovědích NTP serveru. Nejdůležitější údaj `offset` představuje rozdíl mezi referenčním časem a časem lokálních systémových hodin v `ms`, který je zohledněn při měření jednosměrného zpoždění programem D-ITG.

3.1.1 D-ITG

Tento distribuovaný internetový generátor provozu podporuje škálu protokolů na různých úrovních TCP/IP modelu.[1] Mezi jeho možnosti využité v této práci patří právě podpora generování UDP, TCP a VoIP provozu, změna TOS, DS v hlavičce IP paketu, tvorba souborů s naměřenými hodnotami jednosměrného zpoždění, ztrátovosti paketů, kolísání zpoždění a přenosové rychlosti.

Program D-ITG se skládá z několika nástrojů, přičemž v realizaci byly použity ITGSend, ITGRecv a ITGDec. Na vysílací stanici při generování provozu běží vysílací komponenta ITGSend v tzv. skript módu, který umožňuje vytvářet několik paralelních datových přenosů. Jednomu řádku ve skriptu odpovídá jeden tok. Na

[1] Od ledna roku 2011 je oficiálně součástí distribuce Debian.

přijímací stanici je po dobu měření spuštěn nástroj ITGRecv. Po dokončení přenosu ITGDec zajistí analýzu logů, vytvořených programy ITGRecv a ITGSend s výpočtem průměrných hodnot požadovaných veličin. Lze volit průměrné hodnoty za celou dobu provádění experimentu nebo v intervalech definovaných uživatelem.

Příklad postupně zadaných příkazů pro vytvoření a měření dvou VoIP toků z B do A:

Stanice A:
`ITGRecv`

Stanice B:
`ITGSend script_file -x prenos.log & </dev/null`
obsah souboru script_file:
```
-a 192.168.1.13 -rp 10001 -b 0xb8 -t 60000 -m owdm VoIP
-a 192.168.1.13 -rp 10002 -b 0xb8 -t 60000 -m owdm VoIP
```

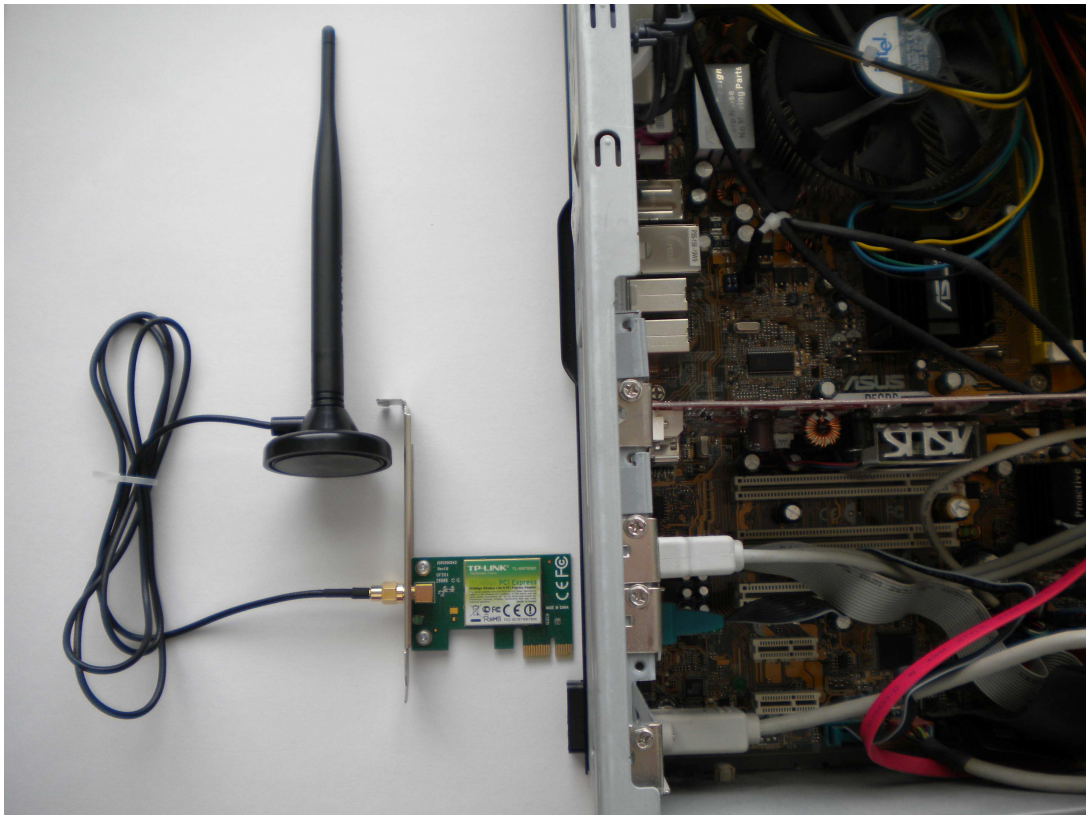
Stanice A:
`ITGDec prenos.log -d 1000 -b 1000 -j 1000 -p 1000`

Nástroj ITGSend čte data ze souboru `script_file`, přepínač `-x` zajišťuje vytvoření logu na přijímací straně s názvem `prenos.log`. Příkaz je spuštěn na pozadí s přesměrovaným standardním vstupem do /dev/null. V souboru `script_file` je přepínačem `-a` definována IP adresa přijímače, `-rp` cílový port, `-b` DiffServ byte jako hexadecimální číslo, `-t` doba vysílání v `ms` a `-m` určuje měření jednosměrného zpoždění. Argument `VoIP` generuje tok s vlastnostmi reálného VoIP hovoru nad transportním UDP protokolem. Případně je možno přepínačem `-x` volit typ kodeku a `-h` definuje typ protokolu. V tomto příkladu i v realizaci jsou ponechány výchozí hodnoty: kodek G.711 s přenosem jednoho vzorku na paket a protokol RTP.

ITGDEC načte data z binárního logu `prenos.log` a vytvoří textové soubory `delay.dat`, `packetloss.dat`, `bitrate.dat` a `jitter.dat` se zprůměrovanými hodnotami po každé 1 `s` měření.

3.1.2 Linuxový směrovač s funkcí přístupového bodu

Tento prvek je vytvořen z klasického stolního PC přidáním bezdrátové karty k již existujícímu FastEthernetovému rozhraní. Zakoupená karta TP-LINK TL-WN781D je vybavena chipsetem Atheros, který jako jeden z mála podporuje režim AP v linuxovém prostředí. Další vlastností je podpora certifikace WME. Poznamenejme,



Obr. 3.2: Karta TP-LINK TL-WN781D před instalací do PCI-E slotu

že v současné době sice běžně dostupný hardware podporuje WME, ne však další funkce z doplňku 802.11e.

Funkčnost karty v režimu AP zajišťuje user space démon hostapd. Následuje výpis řádků s komentáři konfiguračního souboru `hostapd.conf`, ve kterých jsou provedeny změny oproti výchozímu nastavení.

```
interface=wlan0 #jmeno rozhrani
driver=nl80211 #konfiguracni system, zalozeny na netlinku
ssid=shaping #SSID
hw_mode=b #provozni mod
channel=8 #cislo kanalu
wpa=2 #autentizacni a sifrovaci algoritmus WPA2
wpa_passphrase=Jk9+?aH2c #heslo
wpa_key_mgmt=WPA-PSK #sdileny autentizacni klic
wpa_pairwise=CCMP #sifrovaci mechanismus AES
wmm_enabled=1 #podpora WME
```

```
#WME parametry front AP - vysilani ramcu smerem ke klientum
tx_queue_data3_aifs=7 #nizka priorita / AC_BK = pozadi
tx_queue_data3_cwmin=31
tx_queue_data3_cwmax=1023
tx_queue_data3_burst=0
tx_queue_data2_aifs=3 #normalni priorita / AC_BE = best effort
tx_queue_data2_cwmin=31
tx_queue_data2_cwmax=127
tx_queue_data2_burst=0
tx_queue_data1_aifs=1 #vysoka priorita / AC_VI = video
tx_queue_data1_cwmin=15
tx_queue_data1_cwmax=31
tx_queue_data1_burst=6.0
tx_queue_data0_aifs=1 #nejvyssi priorita / AC_VO = hlas
tx_queue_data0_cwmin=7
tx_queue_data0_cwmax=15
tx_queue_data0_burst=3.3

#WME parametry front klientů - vysilani ramcu smerem k AP
wmm_ac_bk_cwmin=5 #nizka priorita / AC_BK = pozadi
wmm_ac_bk_cwmax=10
wmm_ac_bk_aifs=7
wmm_ac_bk_txop_limit=0
wmm_ac_bk_acm=0
wmm_ac_be_aifs=3 #normalni priorita / AC_BE = best effort
wmm_ac_be_cwmin=5
wmm_ac_be_cwmax=7
wmm_ac_be_txop_limit=0
wmm_ac_be_acm=0
wmm_ac_vi_aifs=2 #vysoka priorita / AC_VI = video
wmm_ac_vi_cwmin=4
wmm_ac_vi_cwmax=5
wmm_ac_vi_txop_limit=188
wmm_ac_vi_acm=0
wmm_ac_vo_aifs=2 #nejvyssi priorita / AC_VO = hlas
wmm_ac_vo_cwmin=3
wmm_ac_vo_cwmax=4
wmm_ac_vo_txop_limit=47
wmm_ac_vo_acm=0
```

Pro snazší zatížení Wi-Fi sítě je nastaven standard 802.11b. Číslo kanálu je zvoleno s ohledem na obsazené kanály okolními AP. Parametry WME při vysílání rámců směrem od AP ke klientům a od klientů k AP jsou nakonfigurovány podle doporučení, zajišťující kompatibilitu se standardem 802.11b.

Podporu Wi-Fi zařízení s chipsety Atheros 802.11n v distribuci Debian zajišťuje ovladač ath9k. Jaderný modul je potřeba načíst příkazem:

```
modprobe ath9k
```

Při spouštění démona hostapd se za přepínačem -B definuje umístění konfiguračního souboru:

```
hostapd -B /etc/hostapd/hostapd.conf
```

Posledním krokem je odpovídajícím způsobem nastavit parametry síťových rozhraní v `/etc/network/interfaces` a trvale zajistit funkčnost IP forwardingu volbou `net.ipv4.ip_forward=1` v souboru `/etc/sysctl.conf`. Tímto by měl již být linuxový směrovač s funkcí AP připraven k použití.

3.2 Funkčnost WME

Tato část práce má za cíl prozkoumat rozdíly ve vlastnostech přenosu hlasu při zapnuté a vypnuté službě WME. Dvěma podmínkami pro funkčnost WME jsou:

- WME certifikovaný AP se zapnutou službou WME,
- WME certifikovaná zdrojová i cílová aplikace, popř. aplikace schopná odpovídajícím způsobem označkovat DSCP v hlavičce paketu.

V realizaci je uplatněno značkování podle konvencí: best effortu odpovídá hodnota DSCP 0 a VoIPu 46.

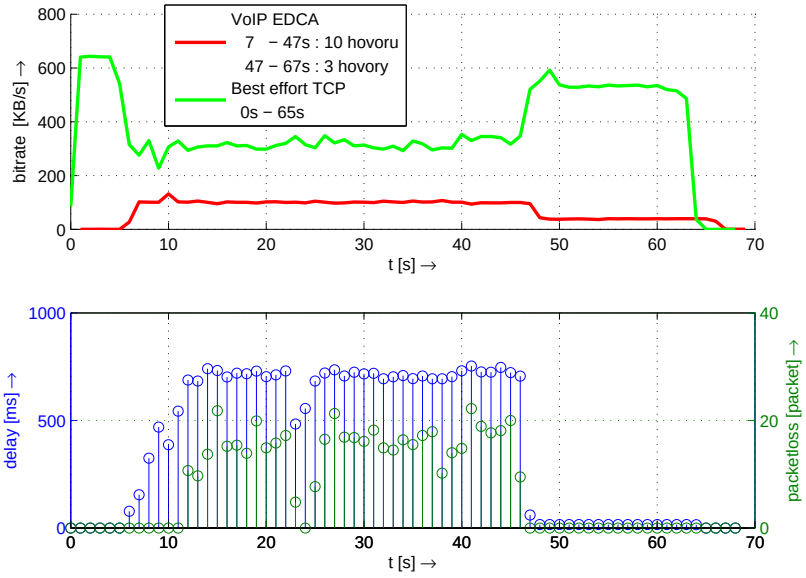
Poznámky ke grafům

Na horním grafu s přenosovou rychlostí je zelená křivka vztažena k provozu mezi stanicemi A a C (spojení pro zatížení AP) a červená křivka s VoIP provozem mezi A a B. Barevné označení koresponduje s obr. 3.1. Spodní graf souvisí s průměrnou ztrátovostí paketů a průměrným zpožděním VoIP provozu mezi A a B jednoho toku.

Přestože program D-ITG umožňuje měřit kolísání zpoždění, v průběhu měření nikdy nedošlo k překročení doporučeného maxima 30 ms na rozdíl od ztrátovosti a zpoždění. Obvyklá hodnota kolísání zpoždění činila 1 ms a jen zcela výjimečně se dostala na 25 ms - pro zachování přehlednosti nebylo kolísání zpoždění do grafů vyneseno. Program D-ITG vysílal pakety jednoho hovoru rychlostí $83,3$ paketů/s. Doporučená max. ztrátovost je 1 %, tedy $0,833$ paketů/s. Pod tuto hranici se většinou podařilo dostat s maximálně třemi navázanými hovory.

3.2.1 WME vypnuto

První měření je posuzováno ve směru upload. V konfiguračním souboru `hostapd.conf` jsou zakomentovány řádky, týkající se WME. Stanice se nyní nemohou registrovat k AP s požadavkem na podporu WME.[2] Z průběhů na obr. 3.3 je patrné, že jednosměrné průměrné zpoždění jednoho z deseti navázaných VoIP spojení v zatížené síti dosahuje nepřijatelně vysokých hodnot - až cca 750 ms, kdežto doporučené maximum je 150 ms. Situace se zlepší po ukončení sedmi z těchto deseti spojení od 47 s.



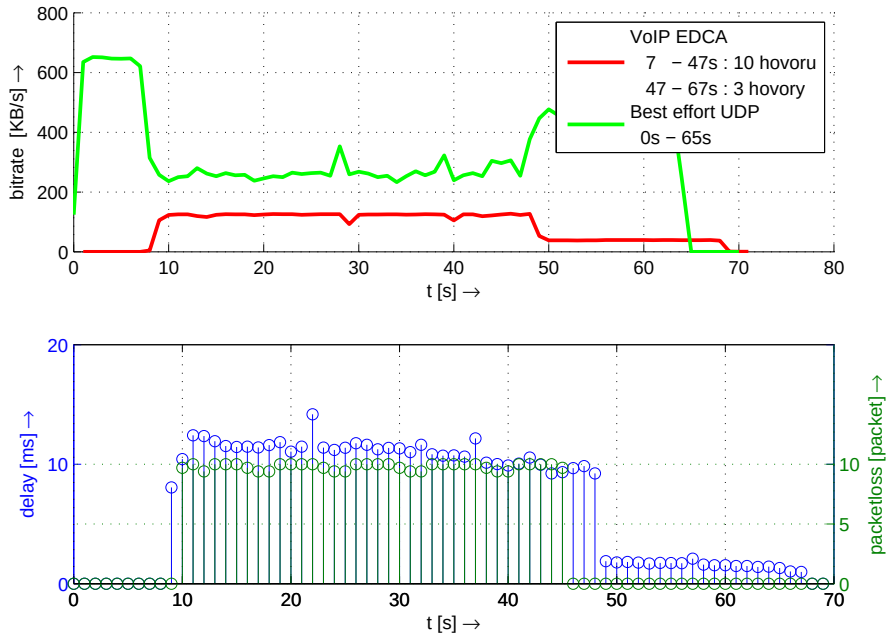
Obr. 3.3: WME vypnuto, upload bez tvarování provozu

3.2.2 WME zapnuto

Odkomentováním řádků konfiguračního souboru `hostapd.conf` s WME parametry pro fronty klientů i AP a restartem démona je povolena funkce WME ve směru upload i download. Ostatní nastavení zůstává shodné s předchozím měřením. Výsledek na obr. 3.4 představuje prokazatelné snížení průměrného zpoždění hovoru ze 750 ms na 10 ms, což je pro VoIP více než dostatečná hodnota. Současně také klesla ztrátovost paketů.

Vzhledem k výraznému zlepšení vlastností jsou všechna následující měření prováděna se zapnutou službou WME.

[2] Dále bylo ověřeno, že pokud je WME na AP zapnuto a veškeré testovací pakety mají nastaveno DSCP na 0, výsledky jsou totožné.



Obr. 3.4: WME zapnuto, upload bez tvarování provozu

3.3 Funkčnost HTB

Jak již bylo uvedeno v teoretickém úvodu v souvislosti s qdisc, na daném rozhraní je možné provádět tvarování pouze pro odchozí provoz, tzv. egress shaping. Příchozí provoz lze omezovat na danou rychlost pomocí zahazování paketů, nikoliv jejich zpožďováním. V některých situacích, např. pokud má směrovač více než dvě síťová rozhraní, vyvstává otázka: kam nastavit globální pravidla, když daný qdisc nelze připojit více jak na jedno rozhraní současně? Nebo pokud by se nejednalo o směrovač, ale o server s jedinou síťovou kartou ve funkci AP, kam nastavit pravidla pro upload, když může být tvarován jen download? K tomuto účelu slouží virtuální síťová rozhraní IMQ a IFB, na něž se přesměruje provoz, který chceme tvarovat a to z jakéhokoliv směru. Jejich porovnání uvádí tab. 3.1.

V našem případě by musel být zaveden tvarovač pro download na wlan0 a další pro upload na eth0. Díky virtuálnímu rozhraní je použit pro upload namísto eth0 ingress na wlan0 s přesměrováním do IFB. Použití virtuálních rozhraní v dalších měřeních tedy není nezbytně nutné, ale shledal jsem je jako užitečné nástroje, a proto je zavedeno IFB.

Tab. 3.1: Porovnání IMQ a IFB

	Výhody	Nevýhody
IMQ	- funguje s NAT - ověřená metoda	- nutnost záplatování jádra - občas nestabilní
IFB	- součástí jádra	- téměř žádná dokumentace - nefunguje s NAT

Kompletní skript s metodou HTB je v příloze A.1. Kostru pro vytvoření hierarchie u downloadu tvoří vybraných osm řádků:

```
tc qdisc add dev wlan0 root handle 1: htb default 11
tc class add dev wlan0 parent 1: classid 1:1 htb rate 450kbps ceil \
   450kbps
tc class add dev wlan0 parent 1:1 classid 1:10 htb rate 67kbps ceil \
   67kbps
tc class add dev wlan0 parent 1:1 classid 1:11 htb rate 383kbps ceil \
   383kbps
tc filter add dev wlan0 protocol ip parent 1: prio 1 u32 match \
   u32 0x00b80000 0x00fc0000 at 0 flowid 1:10
tc filter add dev wlan0 protocol ip parent 1: prio 1 u32 match \
   ip dst 192.168.1.21 flowid 1:10
tc qdisc add dev wlan0 parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev wlan0 parent 1:11 handle 30: pfifo limit 5
```

Nejprve je vytvořen htb root qdisc s identifikátorem 1: na rozhraní wlan0. Přesměrování neklasifikovaného provozu do třídy 1:11 zajišťuje zápis `default 11`. Maximální přenosová rychlost na fyzické vrstvě činí 640 KB/s - viz předchozí měření ověřující funkci WME. Na root qdisc je napojena rodičovská třída s identifikátorem 1:1, která nedovoluje překročení hodnoty downloadu 450 KB/s (parametry `rate` a `ceil`).[3] Stejná hodnota je, viz příloha A.1, definována i pro upload. Pro oba směry celkově platí: $(2 \times 450)\,\mathsf{KB/s} = 900\,\mathsf{KB/s}$.

Směrovač byl zahlcován jednosměrně, a proto by k překročení maximální rychlosti bezdrátového spoje nemělo dojít. Přestože jsem se nesetkal s tím, že by některá z metod v této konfiguraci byla nestabilní, pro nasazení v prostředích, kde by výpadek běhu představoval větší rizika, doporučuji nastavit takové hodnoty přenosových

[3]Nástroj `tc` vyjadřuje kilobyty zkratkou `kbps` a kilobity jako `kbit`.

rychlostí, které v součtu dávají menší než maximálně zjištěnou hodnotu u daného spoje.

Rodičovská třída má dvě listové třídy: 1:10 s 67 KB/s = VoIP, nastaveno dle požadavku přenosové rychlosti pro šest hovorů a 1:11 s 383 KB/s pro zbytek pásma s best effortem, přičemž na každou z nich je zavěšen qdisc pfifo. Ekvivalence parametrů `rate` a `ceil` listových tříd značí, že jde o hierarchii bez půjčování. Jinými slovy, pokud by jedna třída nebyla vytížena maximálně, sourozenec si od ní nevyužitou kapacitu nemůže půjčit.

Klasifikaci provozu provádí filtr u32. Stanici B realizující VoIP hovory lze rozlišit jednak podle cílové ip adresy:

`u32 match ip dst 192.168.1.21 flowid 1:10`,

jednak podle hodnoty DSCP 46 - odpovídá určitým bitům v hlavičce paketu:[4]

`u32 match u32 0x00b80000 0x00fc0000 at 0 flowid 1:10`, kde

- `match u32`: tzv. podmínka shody, následovaná referenční hodnotou a maskou. Z hlavičky je nejprve vybráno 32 bitové slovo, poté maskováno a porovnáno s ref. hodnotou.
- `0x00b80000`: referenční hodnota, odpovídající DSCP 46.
- `0x00fc0000`: maska - udává, které z vybraných 32 bitů hlavičky se mají porovnat s referenční hodnotou.
- `at 0`: offset v bytech od začátku paketu - definuje umístění 32 bitového slova v hlavičce pro extrahování. Jde o volitelnou položku.

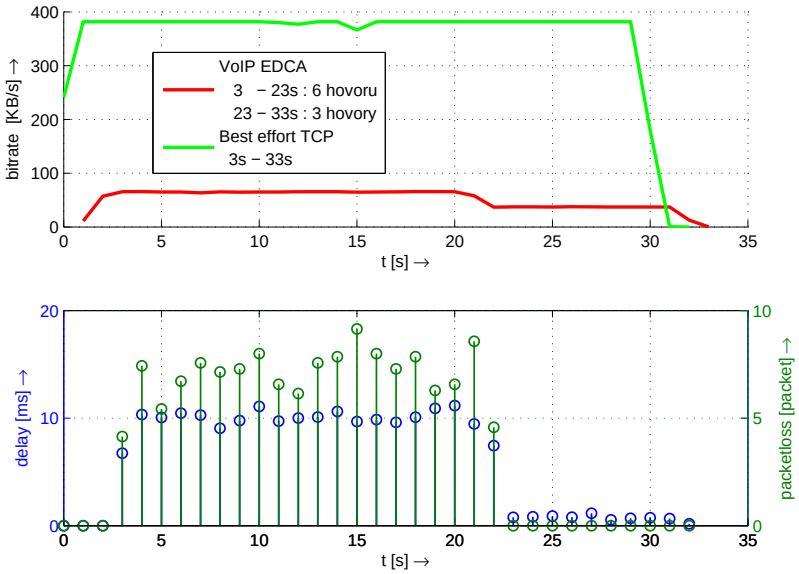
Příklad klasifikace paketů pomocí filtru u32 je uveden v tab. 3.2. Prvních 32 bitů hlavičky je bitově konjugováno s maskou (logický AND). Výsledek je porovnán s referenční hodnotou. V případě shody paket vstupuje do VoIP třídy 1:10.

Tab. 3.2: Prinicip klasifikace paketů pomocí filtru u32

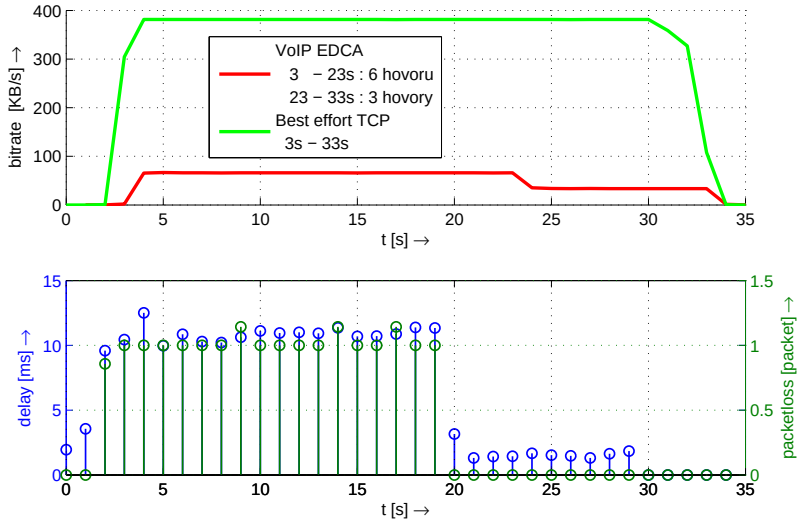
	HEX	BIN
hlavička	46b82341	0100 0110 1011 1000 0010 0011 0100 0001
maska	00fc0000	0000 0000 1111 1100 0000 0000 0000 0000
výsledek	00b80000	0000 0000 1011 1000 0000 0000 0000 0000
reference	00b80000	0000 0000 1011 1000 0000 0000 0000 0000

[4] V praxi se používá zejména způsob klasifikace dle DSCP, pro názornost jsou zde uvedeny dvě možnosti.

Měření ve směru upload na obr. 3.5 potvrzuje správnou funkci metody HTB. Best effort provoz nepřekročí definovaných 383 KB/s a vyhrazených 67 KB/s pro VoIP postačuje. Soutěžení o přístup k médiu bezdrátových stanic neumožňuje snížit průměrnou hodnotu zpoždění šesti hovorů přibližně pod 10 ms. V opačném směru je situace obdobná, pouze dochází ke snížení průměrné ztrátovosti paketů.



Obr. 3.5: HTB upload, bez půjčování



Obr. 3.6: HTB download, bez půjčování

3.4 Funkčnost HFSC

U metody HFSC při zachování naprosto stejných podmínek měření i definicí zpoždění jako v předešlém případě (HTB) nevznikly ve výsledných grafech viditelné rozdíly.[5] Z toho důvodu byla pozornost zaměřena na přesnost tvarované přenosové rychlosti obou metod. Z FTP serveru A byl stahován soubor stanicí C s tvarovanou přenosovou rychlostí na fyzické vrstvě 200 KB/s po dobu 70 s bez jakéhokoliv dalšího provozu. Porovnání metod popisuje tab. 3.3. Z těchto naměřených a vypočtených hodnot je zřetelné, že HFSC je přesnější.

Tab. 3.3: Porovnání přesnosti tvarování přenosové rychlosti pomocí HTB a HFSC

	HTB	HFSC
minimum [KB/s]	177.1	192.3
maximum [KB/s]	202.9	202.9
průměr [KB/s]	198.2	200.2
medián [KB/s]	199.8	199.8
rozptyl [KB/s]	4.862	1.458

3.4.1 HFSC bez definovaného zpoždění

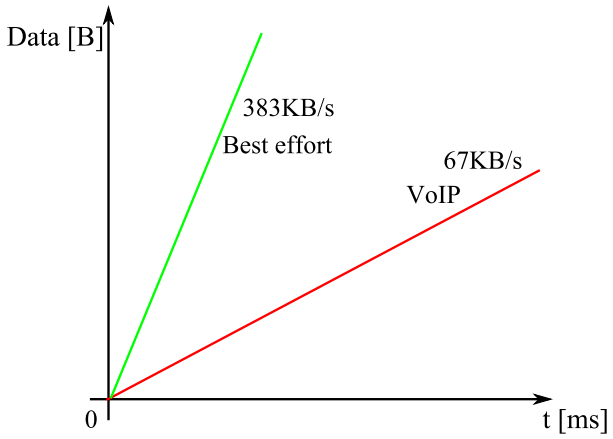
Kompletní skript s metodou HFSC se nachází v příloze A.2. Hierarchie zůstává stejná - jedna rodičovská třída a dvě listové. Pro změnu jde o tvarování provozu s půjčováním. Vybrané řádky slouží pro ukázku změn v zápise vytvoření hierarchie oproti HTB, opět ve směru download.

```
tc qdisc add dev wlan0 root handle 1: hfsc default 11
tc class add dev wlan0 parent 1: classid 1:1 hfsc sc rate 450kbps \
    ul rate 450kbps
tc class add dev wlan0 parent 1:1 classid 1:10 hfsc sc rate 67kbps \
   ul rate 450kbps
tc class add dev wlan0 parent 1:1 classid 1:11 hfsc sc rate 383kbps \
   ul rate 450kbps
```

[5] V obou případech šlo o tvarování provozu bez půjčování.

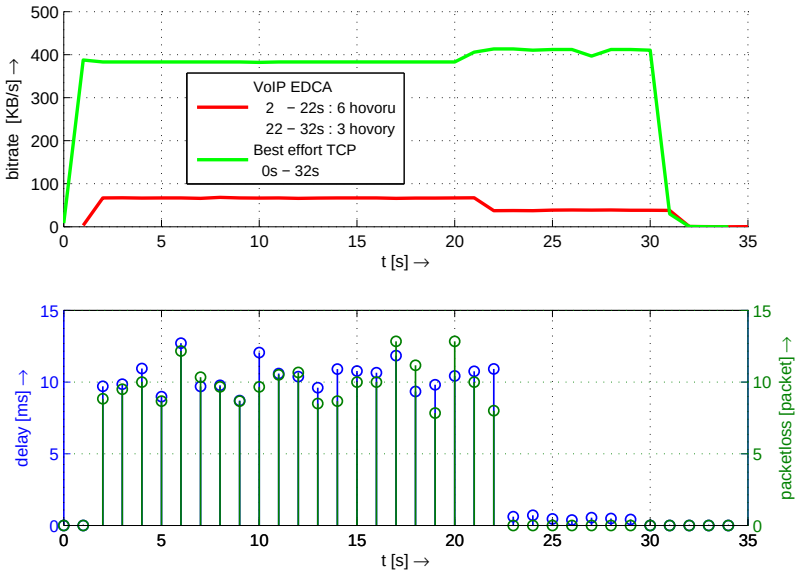
Na rozhraní je zavěšen root qdisc s identifikátorem handle 1:, následuje název metody `hfsc` s určením výchozí třídy 1:11 pro všechen provoz, který není klasifikovatelný filtry. Na qdisc je připojena rodičovská třída, jejíž potomky jsou dvě listové třídy. V porovnání s HTB je v tomto zápise parametr `sc rate` ekvivalentem `rate`, a `ul rate` ekvivalentem `ceil`. Pro vyzkoušení půjčování nevyužité šířky pásma byla změněna hodnota `ul rate` na maximální jednosměrnou přenosovou rychlost linky.

Výše uvedeným způsobem se definují lineární křivky obsluhy, které graficky znázorňuje obr. 3.7.

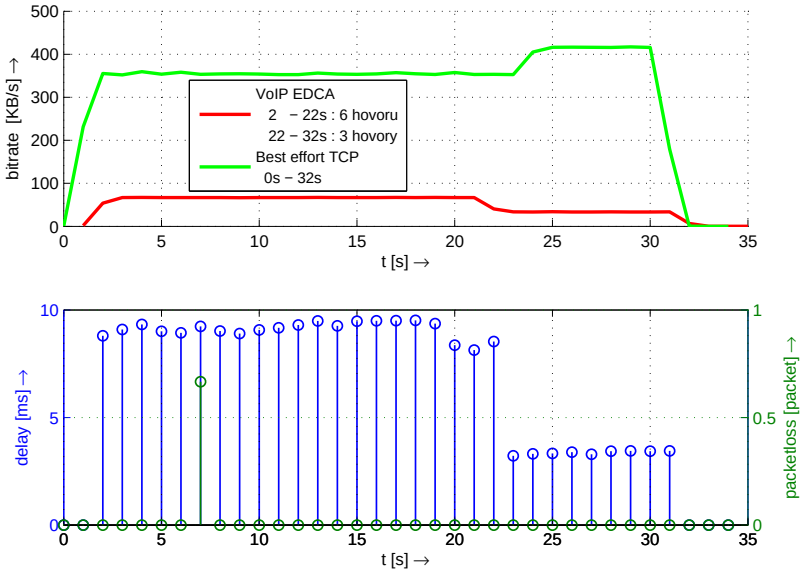


Obr. 3.7: HFSC, lineární křivky obsluhy listových tříd

Grafy 3.8, 3.9 potvrzují správnou funkci metody. V čase 22 s si best effort půjčuje nevyužitou kapacitu od třídy pro VoIP tak, aby nedošlo k překročení maximální povolené rychlosti 450 KB/s. U downloadu při ukončení tří hovorů dochází před vypůjčením uvolněné kapacity sourozeneckou třídou k časové hysterezi 2 s.



Obr. 3.8: HFSC upload, lineární křivky obsluhy



Obr. 3.9: HFSC download, lineární křivky obsluhy

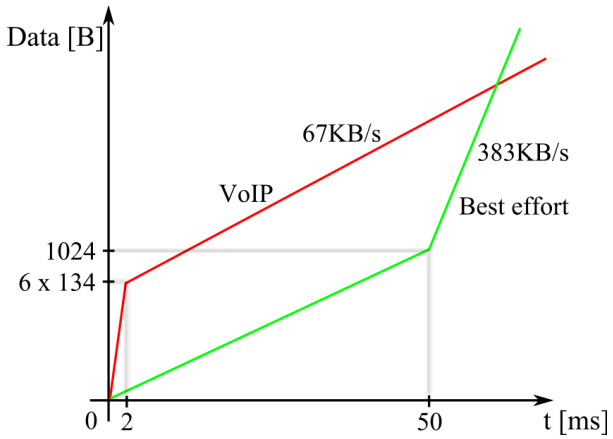
3.4.2 HFSC s definovaným zpožděním

Jestliže chceme kromě přenosové rychlosti určit také maximální přípustné zpoždění např. pro multimediální aplikace, je nutné zavést po částech lineární křivky obsluhy. Řekněme, že potřebujeme snížit zpoždění VoIPu na 2 ms. Analýzou provozu bylo zjištěno, že program D-ITG simuluje jeden přenos hlasu vysíláním rámců o velikosti

134 B. Při současných šesti přenosech lze počítat s 804 B. V definici listové třídy se nastaví `umax 804b` a `dmax 2ms`. Tímto je vytvořena první ze dvou částí křivky obsluhy pro VoIP. Sklon druhé části o 67 KB/s se definuje výrazem `rate 67kbps`. Maximální přípustnou rychlost vypůjčenou od rodiče určuje `ul rate`. Tímto je nakonfigurována první po částech lin. křivka obsluhy pro VoIP, na obr. 3.10 vyznačena červeně.

První část křivky obsluhy best effortu je odvozena ze zbývající kapacity linky. Z celkové přenosové rychlosti 450 KB/s může VoIP zabrat $\frac{804}{2 \cdot 10^{-3}} = 402$ KB/s a nevyužito tak zůstává 48 KB/s. Obvyklá velikost rámce simulace best efforu činí 1024 B. Dosažitelná minimální přenosová rychlost je potom $\frac{1024}{48 \cdot 10^3} = 21,333$ ms. Jelikož tento druh provozu není náchylný na hodnotu zpoždění, pro odzkoušení byla zvolena hodnota `dmax 50ms`. Sklon druhé části křivky obsluhy definuje `rate 383kbps`. Součet obou přenosových rychlostí druhých částí by neměl překročit 450 KB/s.

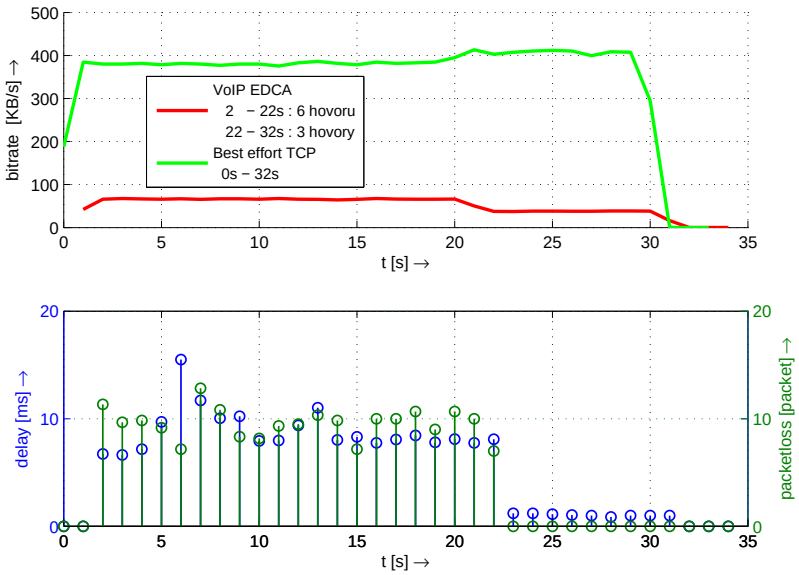
```
tc class add dev wlan0 parent 1:1 classid 1:10 hfsc sc umax 804b \
   dmax 2ms rate 67kbps ul rate 450kbps
tc class add dev wlan0 parent 1:1 classid 1:11 hfsc sc umax 1042b \
   dmax 50ms rate 383kbps ul rate 450kbps
```



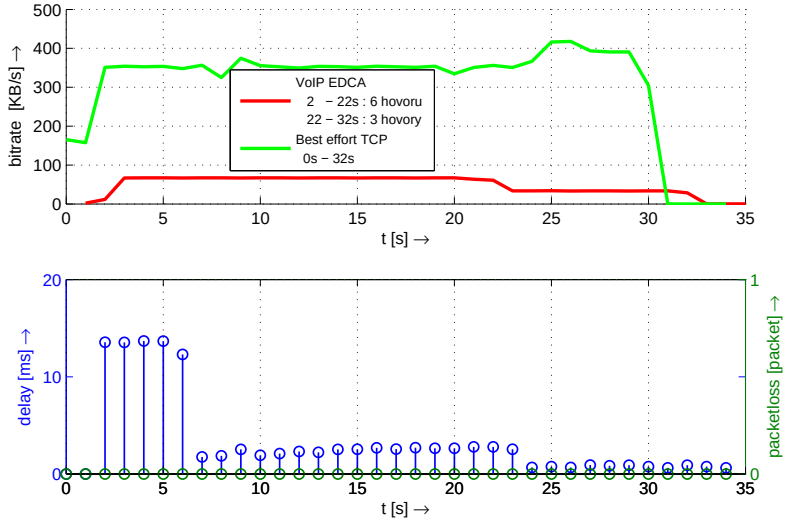
Obr. 3.10: HFSC, po částech lineární křivky obsluhy listových tříd

Zavedení po částech lineárních křivek obsluhy ve směru upload nepřineslo požadované snížení zpoždění - viz graf 3.11. Vysvětlení spočívá v tom, že ještě před příchodem paketů do qdiscu na rozhraní vzniká zpoždění soutěžením stanic o přístup k médiu, které již dále nelze ovlivnit. Tvarování provozu na určitou rychlost fungovalo bez problémů.

Ve směru download jsou pakety odesílány z AP k bezdrátovým stanicím až po tvarování. V této situaci již bylo zpoždění sníženo na nastavenou hodnotu. Mezi uvolněním části kapacity linky ukončením tří ze šesti hovorů a její výpůjčkou sourozenecké třídě opět vnikla přibližně dvousekundová časová hystereze. Stejně tak žádaného snížení zpoždění bylo dosaženo po několika sekundách od začátku navázání VoIP spojení.



Obr. 3.11: HFSC upload, po částech lineární křivky obsluhy



Obr. 3.12: HFSC download, po částech lineární křivky obsluhy

4 ZÁVĚR

Náplní teoretické části práce byl popis Linuxových nástrojů a metod pro zajištění QoS. Metoda HTB byla zvolena vzhledem k deklarované vyšší přesnosti a transparentnější konfiguraci oproti klasické CBQ. Druhá metoda HFSC kromě určování přenosové rychlosti šířky pásma navíc umožňuje definovat maximální zpoždění, což lze s výhodou použít např. u přenosu hlasu a videa.

V simulátoru NCTUns bylo ověřeno chování bezdrátové sítě 802.11e. Simulace ukázala, že v zahlcené síti může dojít ke stavu, kdy jsou přenosy hlasu využívající přístupovou metodu EDCA potlačeny na úkor jiných stejně důležitých přenosů využívající přístupovou metodu HCCA. Z toho důvodu by pro zajištění QoS měla být lepší volbou HCCA.

V praktické části práce došlo k upravení klasického PC s GNU/Linux Debian Squeeze na směrovač spolu s funkcí přístupového bodu a podporou služby WME. Následně byl zjištěn pozitivní vliv této služby na kvalitu přenosu datových toků. Ukázalo se, že metody pro tvarování provozu a WME jsou dvě odlišné záležitosti, které se mohou vzájemně doplňovat.

Tvarovače byly navrženy k použití na sítích typu Ethernet a ke své funkci potřebují znát přenosovou rychlost linky. V bezdrátových sítích tato rychlost značně kolísá v závislosti na více faktorech, a proto je dobré při konfiguraci počítat s nižšími hodnotami. Pokud by byla definována vyšší tvarovaná rychlost než reálně dosažitelná, nemusel by tvarovač správně fungovat. Možným řešením tohoto problému by byla implementace tvarování na MAC podvrstvě linkové vrstvy.

Vyšší přesnost v měření tvarované přenosové rychlosti prokázalo HFSC oproti HTB. Dále byly vyzkoušeny varianty s půjčováním i bez půjčování od rodičovských tříd. HFSC s po částech lineární křivkou obsluhy fungovalo ve směru download. Ve směru upload již nešlo snížit definované zpoždění, jelikož tvarovač nemohl ovlivnit zpoždění způsobené především soutěžením o přístup k médiu stanicemi v zahlcené síti.

LITERATURA

[1] GHEORGHE, Lucian. *Designing and Implementing Linux Firewalls nad QOS using netfilter, iproute2, NAT, and L7-filter.* Birmingham : Packt Publishing, 2006. 272 s. ISBN 1-904811-65-5.

[2] XIPENG, Xiao. *Technical, Commercial and Regulatory Challenges of QOS, An Internet Service Model Perspective.* Burlington, USA : Morgan Kaufmann Publishers, 2008. 274 s. ISBN 978–0–12–373693–2.

[3] Ubik, Sven. *Technická zpráva TEN-155 CZ číslo 6/2000 QoS a diffserv - Úvod do problematiky* 2000. 12s

[4] Hubert, Bert. *Linux Advanced Routing and Traffic Control HOWTO* 2003. 152s

[5] Differentiated Service Theory. *Differentiated Service on Linux HOWTO* [online]. [cit. 2010-11-11]. Dostupné z WWW: `http://www.opalsoft.net/qos/DS.htm`.

[6] BROWN, Martin. *Traffic Control HOWTO* [online]. 2006 [cit. 2010-11-11]. Dostupné z WWW: `http://tldp.org/HOWTO/Traffic-Control-HOWTO/index.html`.

[7] Devera, Martin. *Princip a užití HTB QoS disciplíny* 2002. 10s [cit. 2010-11-15] Dostupné z WWW: `http://luxik.cdi.cz/~devik/qos/htb/cs/slt02htb.ps`

[8] Devera, Martin. *Hierachical token bucket theory* [online]. 2002 [cit. 2010-11-15]. Dostupné z WWW: `http://luxik.cdi.cz/~devik/qos/htb/manual/theory.htm`

[9] GARROPPO, R.G. et al. *A comparison of HTB based Channel-Aware Schedulers for 802.11 systems*[online]. Department of Information Engineering, University of Pisa, 2005 [cit. 2010-11-21]. Dostupné z WWW: `http://twelve.unitn.it/docs/PI-1.pdf`

[10] WANG, Shie Yuan; CHOU, Chih-Liang; LIN, Chih-Che. *The GUI User Manual for the NCTUns 6.0 Network Simulator and Emulator* [online]. Network and System Laboratory, Department of Computer Science, National Chiao Tung University, Taiwan : 2010 [cit. 2010-11-27]. Dostupné z WWW: `http://nsl10.cs.nctu.edu.tw/`

[11] MOLNÁR, Karol. *Zajištění kvality služeb v bezdrátových a mobilních sítích* [online]. [cit. 2010-12-05]. Dostupné z WWW: `http://www.utko.feec.vutbr.cz/~molnar/mmos/wifi_umts.pdf`

[12] STOICA, Ion; ZHANG, Hui; EUGENE NG, T.S. *A Hierarchical Fair Service Curve Algorithm for Link-Sharing, Real-Time and Priority Services.* School of Computer Science, Carneige Mellon University, PA 15213, 14s.

[13] RECHERT, Klaus; MCHARDY, Patrick. *HFSC Scheduling with Linux.* [online]. 2005 [cit. 2010-12-08]. Dostupné z WWW: `http://linux-ip.net/articles/hfsc.en/.`

[14] Wi-Fi Alliance. *Wi-Fi CERTIFIED^{TM} for WMM^{TM} - Support for Multimedia Applications with Quality of Service in Wi-Fi® Networks* [online]. 2004, [cit. 2011-4-24]. Dostupné z WWW: `http://www.wi-fi.org/files/wp_1_WMM%20QoS%20In%20Wi-Fi_9-1-04.pdf`

[15] Turek, Lukáš. *Řízení toku v přístupových bodech bezdrátové sítě IEEE 802.11* [online]. 2009, [cit. 2011-4-24]. Dostupné z WWW: `http://8an.praha12.net/papers/RizeniTokuWiFi.pdf`

SEZNAM SYMBOLŮ, VELIČIN A ZKRATEK

AC	řízení přístupu – *Admission Control*
AC	kategorie přístupu –*Access Cathegory*
AP	přístupový bod – *Access Point*
BA	*Behavior Aggregate*
CW	okno soutěžení – *Contention Window*
DCF	distribuovaná koordinační funkce – *Distributed Coordination Function*
DiffServ	diferencované služby – *Differentiated Services*
DITG	*Distributed Internet Traffic Generator*
DSCP	*Differentiated Service Code Point*
EDCA	rozšířený distribuovaný přístup ke kanálu – *Enhanced Distributed Channel Access*
FSCLS	*Fair Service Curve Link Sharing*
HCCA	přístup ke kanálu řízený HCF – *HCF - Controlled Channel Access*
HCF	hybridní koordinační funkce – *Hybrid Coordination Function*
HFSC	*Hierarchical Fair Service Curve*
IFB	*Intermediate Functional Block*
IMQ	*Intermediate Queueing Device*
IntServ	integrované služby – *Integrated Services*
MF	*Multi Field*
NAPT	překlad portů a síťových adres – *Network Address and Protocol Translation*
NAT	překlad síťových adres – *Network Address Translation*
NTP	*Network Time Protocol*
PCF	centralizovaná koordinační funkce – *Point Coordination Function*
QoS	kvalita služeb – *Quality Of Service*

RSVP *Resource Reservation Protocol*

TCA *Traffic Conditioning Agreement*

ToS *Type Of Service*

TXOP *příležitost přenosu - Transmission Oportunity*

WLAN bezdrátová lokální síť – *Wireless Local Area Network*

WME *Wireless MultiMedia Extensions*

WMM *WiFi MultiMedia*

A SKRIPTY PRO TVAROVÁNÍ PROVOZU

A.1 HTB

```
#!/bin/bash
# chkconfig: - 11 89
# description: HTB + IFB script

DD=wlan0
UD=ifb0
i=1

start () {
echo "Starting HTB.."
tc qdisc add dev $DD root handle 1: htb default 11

tc class add dev $DD parent 1: classid 1:1 htb rate 450kbps ceil 450kbps
tc class add dev $DD parent 1:1 classid 1:10 htb rate 67kbps ceil 67kbps
tc class add dev $DD parent 1:1 classid 1:11 htb rate 383kbps ceil 383kbps

tc filter add dev $DD protocol ip parent 1: prio 1 u32 match \
   u32 0x00b80000 0x00fc0000 at 0 flwid 1:10
tc filter add dev $DD protocol ip parent 1: prio 1 u32 match \
   ip dst 192.168.1.21 flowid 1:10
tc qdisc add dev $DD parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev $DD parent 1:11 handle 30: pfifo limit 5
echo "Downlink OK"

modprobe ifb
ip link set dev $UD up

tc qdisc add dev $DD ingress
tc filter add dev $DD parent ffff: protocol ip prio 10 u32 \
   match u32 0 0 flowid 1:1 \
   action mirred egress redirect dev $UD
tc qdisc add dev $UD root handle 1: htb default 11

tc class add dev $UD parent 1: classid 1:1 htb rate 450kbps ceil 450kbps
tc class add dev $UD parent 1:1 classid 1:10 htb rate 67kbps ceil 67kbps
```

```
tc class add dev $UD parent 1:1 classid 1:11 htb rate 383kbps ceil 383kbps

tc filter add dev $UD protocol ip parent 1: prio 1 u32 match \
   u32 0x00b80000 0x00fc0000 at 0 flowid 1:10
tc filter add dev $UD protocol ip parent 1: prio 1 u32 match \
   ip src 192.168.1.21 flowid 1:10

tc qdisc add dev $UD parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev $UD parent 1:11 handle 30: pfifo limit 5
echo "Uplink OK"
}
stop () {
echo -n "Stopping HTB.."
tc qdisc del dev $DD root
tc qdisc del dev $UD root
tc qdisc del dev $DD ingress 2>/dev/null
echo OK
}

case "$1" in
start)
start
;;
stop)
stop
;;
)
echo "Usage: htb.init {start|stop}"
RETVAL=1
esac
exit $RETVAL
```

A.2 HFSC

```
#!/bin/bash
# chkconfig: - 11 89
# description: HFSC + IFB script

DD=wlan0
UD=ifb0
i=1

start () {
echo "Starting HFSC.."
tc qdisc add dev $DD root handle 1: hfsc default 11
tc class add dev $DD parent 1: classid 1:1 hfsc sc rate 450kbps \
   ul rate 450kbps

#A) download shaping s definici zpozdeni
tc class add dev $DD parent 1:1 classid 1:10 hfsc sc umax 804b \
   dmax 2ms rate 67kbps ul rate 450kbps
tc class add dev $DD parent 1:1 classid 1:11 hfsc sc umax 1042b \
   dmax 50ms rate 383kbps ul rate 450kbps

#B) upload shaping bez definice zpozdeni
#tc class add dev $DD parent 1:1 classid 1:10 hfsc sc rate 67kbps \
   ul rate 450kbps
#tc class add dev $DD parent 1:1 classid 1:11 hfsc sc rate 383kbps \
   ul rate 450kbps

tc filter add dev $DD protocol ip parent 1: prio 1 u32 match \
   u32 0x00b80000 0x00fc0000 at 0 flowid 1:10
tc filter add dev $DD protocol ip parent 1: prio 1 u32 match \
   ip dst 192.168.1.21 flowid 1:10
tc qdisc add dev $DD parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev $DD parent 1:11 handle 30: pfifo limit 5

modprobe ifb
ip link set dev $UD up
```

```
tc qdisc add dev $DD ingress
tc filter add dev $DD parent ffff: protocol ip prio 10 u32 \
   match u32 0 0 flowid 1:1 \
   action mirred egress redirect dev $UD
tc qdisc add dev $UD root handle 1: hfsc default 11
tc class add dev $UD parent 1: classid 1:1 hfsc sc rate 450kbps \
   ul rate 450kbps

#A) upload shaping s definici zpozdeni
tc class add dev $UD parent 1:1 classid 1:10 hfsc sc umax \
   804b dmax 2ms rate 67kbps ul rate 450kbps
tc class add dev $UD parent 1:1 classid 1:11 hfsc sc umax \
   1042b dmax 50ms rate 383kbps ul rate 450kbps

#B) upload shaping bez definice zpozdeni
#tc class add dev $UD parent 1:1 classid 1:10 hfsc sc rate 67kbps \
   ul rate 450kbps
#tc class add dev $UD parent 1:1 classid 1:11 hfsc sc rate 383kbps \
   ul rate 450kbps

tc filter add dev $UD protocol ip parent 1: prio 1 u32 match \
   u32 0x00b80000 0x00fc0000 at 0 flowid 1:10
tc filter add dev $UD protocol ip parent 1: prio 1 u32 match \
   ip src 192.168.1.21 flowid 1:10
tc qdisc add dev $UD parent 1:10 handle 20: pfifo limit 5
tc qdisc add dev $UD parent 1:11 handle 30: pfifo limit 5
echo OK
}

stop () {
echo -n "Stopping HFSC.."
tc qdisc del dev $DD root
tc qdisc del dev $UD root
tc qdisc del dev $DD ingress 2>/dev/null
echo OK
}

case "$1" in
start)
```

```
start
;;
stop)
stop
;;
)
echo "Usage: hfsc.init {start|stop}"
RETVAL=1
esac

exit $RETVAL
```