VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OMEZENÍ PROVOZU PEER-TO-PEER SÍTÍ

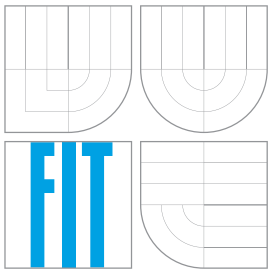
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN ŠÍPOŠ

BRNO 2010

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV POČÍTAČOVÝCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF COMPUTER SYSTEMS

OMEZENÍ PROVOZU PEER-TO-PEER SÍTÍ

REDUCING PEER-TO-PEER NETWORK TRAFFIC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN ŠÍPOŠ

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. MARTIN ŽÁDNÍK

BRNO 2010

Abstrakt

Bakalárska práca sa zaoberá detekovaním peer-to-peer sietí, ich sledovaním a obmedzovaním. Za týmto účelom bola vytvorená aplikácia, ktorá, pomocou voľne dostupných nástrojov, je schopná tieto siete detekovať, sledovať a obmedzovať na úrovni aplikačnej vrstvy, aj na úrovni blokovania portov. Aplikácia bola vytvorená v jazyku Java a je určená pre operačný systém Linux.

Abstract

Bachelor thesis deals with peer-to-peer detection watching and reducing bandwith. An application was created for this purpose that is able to detect watch and control such networks at application layer or to block specified ports using opensource utilities. The application was created in Java language and is dedicated to Linux operating system.

Klíčová slova

p2p, omezení, provoz, l7-filter, síťová komunikace

Keywords

p2p, reducing, traffic, l7-filter, network traffic, traffic control

Citace

Martin Šípoš: Omezení provozu peer-to-peer sítí, bakalářská práce, Brno, FIT VUT v Brně, 2010

Omezení provozu peer-to-peer sítí

Prohlášení

Prohlašuji, že jsem tuto bakalářskou práci vypracoval samostatně pod vedením Ing. Martina Žádníka. Uvedl jsem všechny literární prameny a publikace, ze kterých jsem čerpal.

.....
Martin Šípoš
19.05.2010

Poděkování

Týmto by som chcel poďakovať vedúcemu práce Ing. Martinovi Žádníkovi za trpezlivosť, odbornú pomoc a poskytnuté konzultácie.

Obsah

Kapitola 1

Úvod

V posledných rokoch nastal veľký nárast používania peer-to-peer sietí. Tieto siete vďaka veľmi veľkému využívanému prenosovému pásmu dokážu spôsobiť, že dôležitejšie služby (web, dns, mail a iné) sa stanú nedostupnými. Vážnosť problému narastá v sieťach, ktoré nemajú príliš veľkú prenosovú rýchlosť.

Detekcia peer-to-peer sietí nám ponúka lepší prehľad o spôsobe využitia našej siete. Pokiaľ zistíme, že vďaka týmto sieťam nemôžme garantovať dostupnosť iných služieb, je vhodné peer-to-peer siete obmedziť.

Cieľom tejto práce je navrhnúť systém automatickej detekcie peer-to-peer sietí a obmedzenia ich sieťovej premávky. Následne implementovať tento návrh a vytvoriť aplikáciu, ktorá po umiestnení na router internetového poskytovateľa, dokáže tieto siete úspešne nájsť a redukovať ich prenosové pásmo. Štatistiky o jednotlivých tokoch v sieti bude možné sledovať pomocou grafov a zároveň bude možné tieto štatistiky ukladať do databáze na prípadnú neskoršiu analýzu.

Následujúci text je rozdelený do kapitol, ktorých obsah je následovný. V druhej kapitole práca popisuje základné informácie o sieťovej komunikácii, venuje sa popisu niektorých najznámejších peer-to-peer protokolom. Na záver kapitoly sú vysvetlené najpoužívanejšie typy detekcie peer-to-peer sietí. Tretia kapitola sa zaoberá nástrojmi, ktoré boli použité pre správny chod môjho programu. Je vysvetlený spôsob, akým tieto programy pracujú, a ako ich treba správne konfigurovať. Štvrtá kapitola popisuje mnou navrhnutý systém. Piata kapitola popisuje implementáciu jednotlivých častí systému. Šiesta kapitola obsahuje pokusy mnou vytvorenej aplikácie. Sú popísané obmedzenia, na ktoré som prišiel počas testovania a pomocou grafov sú znázornené výsledky detekcie. Posledná, siedma kapitola je záver, v ktorej je vyhodnotená aplikácia. Sú v nej uvedené hlavné nedostatky a návrh na vylepšenie aplikácie.

Kapitola 2

Teoretický rozbor

V praxi existujú 2 základné typy sieťovej komunikácie: komunikácia klient - server a komunikácia peer-to-peer. Využívané sú oba tieto typy v rôznych aplikáciách.

2.1 Klient - server komunikácia

V tomto type komunikácie vystupujú 2 základné subjekty: server a klient. Každý z nich zohráva v komunikácii inú úlohu. Iniciátorom spojenia je klientská aplikácia. Otvorí si na počítači voľne dostupný port a snaží sa nadviazať spojenie so vzdialeným serverom. Za týmto účelom vytvorí patričné pakety[1] a odošle ich na IP adresu, na ktorej by mala byť spustená serverová aplikácia. Pokiaľ je serverová aplikácia spustená, je dostupná na určitej, klientovi vopred známej, IP adrese a porte. Po prijatí správy od klienta je táto správa spracovaná a server na ňu patričným spôsobom zareaguje. Klientovi odpovie odoslaním požadovaných informácií. Takýto typ komunikácie si môžeme pozrieť na Obrázku 2.1.

Nevýhoda tohoto spojenia je, že pokiaľ sa na server pripojí príliš veľké množstvo klientov, ktorých požiadavky na server vyžadujú značný výpočtový čas, server môže mať problémy so spracovávaním a komunikovaním s klientmi. Tento negatívny fakt má za následok zníženie prenosových rýchlostí serveru, spomalenie odozvy klientom, tzv. zamrznutie serveru, ba môže viesť až k úplnému zlyhaniu aplikácie alebo celého počítača.

Tento typ komunikácie je veľmi ľahko identifikovateľný z dôvodu fixných IP adries / portov. Z tohoto dôvodu nemusíme pri detekovaní používať zložité algoritmy a pre nás je táto komunikácia menej podstatná.

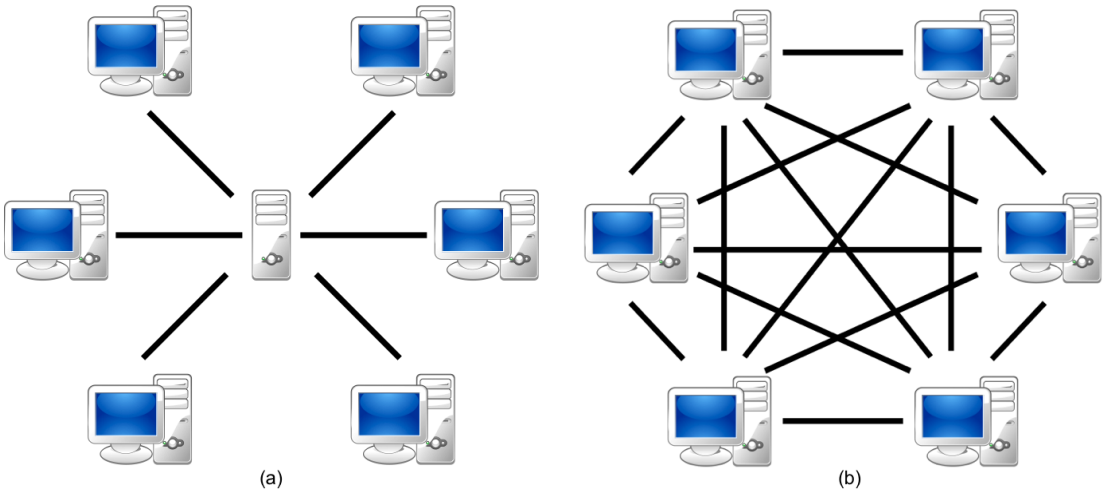
2.2 Peer-to-peer komunikácia

Oproti komunikácii typu klient - server nám v komunikácii vystupuje iba 1 základný prvok tzv. peer[2] (ďalej budem používať slovenský výraz *uzol*), ktorý plní funkciu aj servera, aj klienta. Dáta sú uložené u každého uzlu, ktorý sa do tejto siete pripája. Takýto typ aplikácie začne na počítači otvárať viaceré porty a hľadá ďalšie uzly, na ktoré sa môže pripojiť. Po nájdení uzlu sa prenesú dáta o iných uzloch v sieti, ktoré môžu byť následne kontaktované.

Každý klient si udržiava databázu dostupných uzlov. Pri vyhľadávaní dát kontaktuje okolité uzly a zisťuje, či sa u nich nachádzajú požadované dáta (viď. Obrázok 2.1). Tie sú

[1] paket - jednotka dát prenášaných prostredníctvom počítačovej siete

[2] peer - slovensky uzol má význam seberovný, rovesník - z toho vzniklo peer-to-peer (rovný s rovným)



Obrázek 2.1: Ukážka typu sietí: (a) klient server a (b) peer-to-peer (prevzaté z [7]).

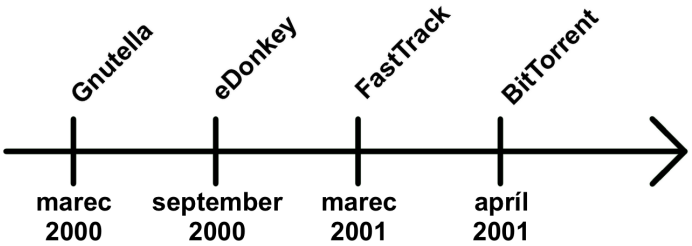
následne sťahované buď z jedného zdroja alebo z viacerých. Vďaka tomu peer-to-peer siete netrpia preťažovaním počítačových staníc.

Takéto siete prinášajú veľa výhod, a preto zaznamenali veľký rozmach a v súčasnosti sú veľmi používané. Navzdory dobrým vlastnostiam peer-to-peer sietí sa kladie čoraz väčší dôraz na ich detekciu, obmedzovanie, dokonca až blokovanie. Dôvodom je, že tieto siete musia medzi sebou neustále komunikovať, čo sa prejavuje na veľkosti prenesených dát. Vďaka tomu využívajú široké prenosové pásmo. Bohužiaľ neraz zaberajú tak široké prenosové pásmo, že nemôže v sieti prebiehať iná (dôležitejšia) komunikácia. Toto správanie je nežiadúce, a preto sa rôzni ISP[3] rozhodli obmedzovať peer-to-peer aplikácie.

Úplné blokovanie peer-to-peer komunikácie nie je najlepšia voľba, ako sa vyhnúť tejto komunikácii. Úplným blokovaním sa dosiahne toho, že vývojári peer-to-peer aplikácií si dajú väčšiu námahu na vývoj takých spôsobov komunikácie, aby boli horšie detekované. Cieľom ISP by preto nemalo byť úplné blokovanie týchto sietí, ale obmedzenie ich sieťovej premávky na prijateľnú hodnotu. Tá by mala byť prispôsobená prenosovému pásmu linky ISP aj koncovému užívateľovi.

2.3 Peer-to-peer protokoly

Historický vývoj popisovaných peer-to-peer protokolov viď Obrázok 2.2.



Obrázek 2.2: Historický vývoj peer-to-peer protokolov (podľa [5]).

[3]ISP - *Internet Service Provider* je akákoľvek organizácia poskytujúca pripojenie k internetu [12]

2.3.1 Gnutella

Informácie ohľadom tohto protokolu sú čerpané z publikácie [6]. Gnutella je kompletne distribuovaný protokol[4]. V sieti Gnutella je každý klient serverom a naopak, preto sa klient nazýva servent. Po pripojení do siete servent hľadá ostatných serventov v sieti. Po úspešnom nájdení pomocou gnutella protokolu komunikuje s ostatnými serventmi v sieti. Vytvorenie TCP spojenia má následovný formát:

```
GNUTELLA CONNECT/<verzia protokolu>\n\n
```

Odpoveď na vytvorenie spojenia:

```
GNUTELLA OK\n\n
```

Žiadosť klienta na zaslanie súboru:

```
GET /get/<index súboru>/<názov súboru>
/HTTP/1.0 \r \n
Connection: Keep-Alive\r\n
Range: byte=0-\r\n
User-Agent: <Meno>\r\n
\r\n
```

Odpoveď na žiadosť súboru má následovný tvar:

```
HTTP 2000 OK\r\n
Server: <Meno>\r\n
Content-type: \r\n
Content-length: \r\n
\r\n
```

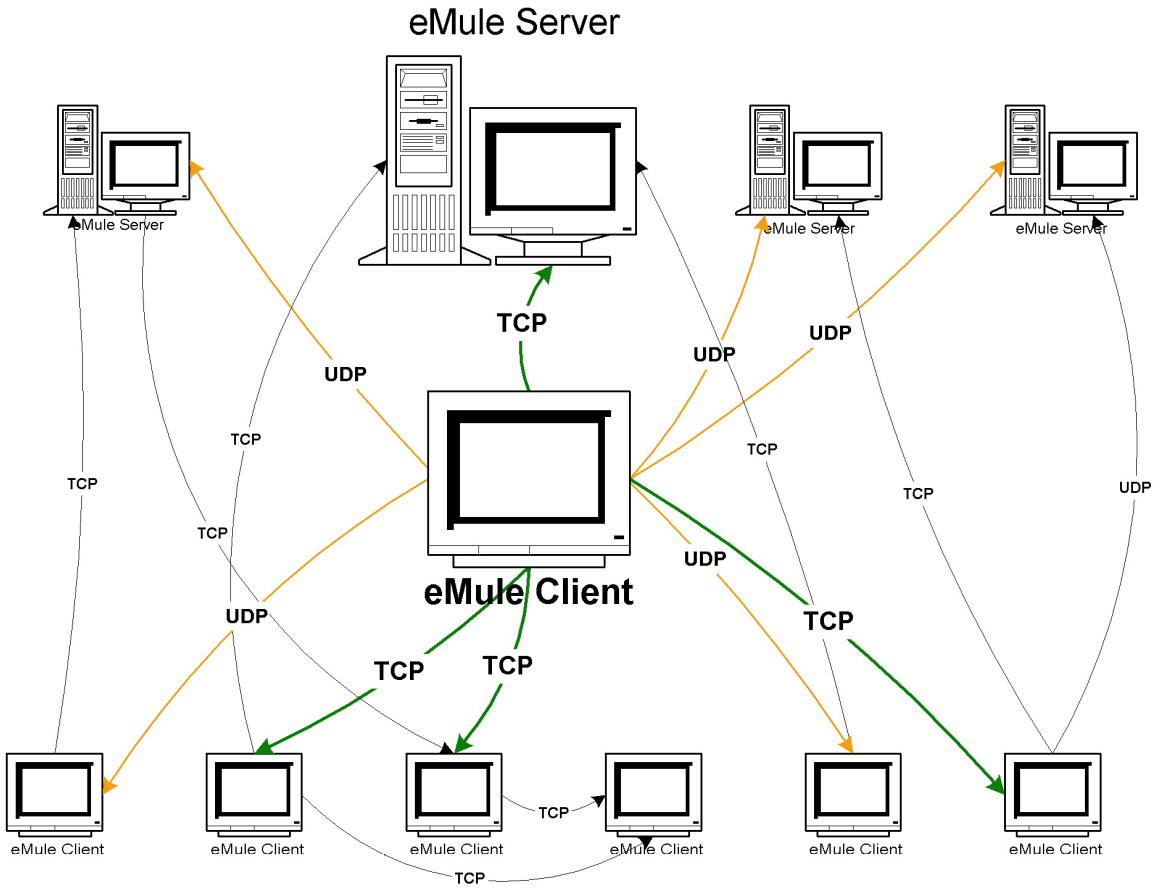
Na základe týchto informácií môžeme priradiť daný tok protokolu Gnutella, ak:

- Reťazec nasledujúci za TCP/IP hlavičkou je `GNUTELLA`, `GET` alebo `HTTP`.
- Pokiaľ je prvý reťazec `GET` alebo `HTTP`, musí ho nasledovať:
  - `User-Agent: <Meno>`
  - `UserAganet: <Meno>`
  - `Server: <Meno>`

  pričom `<Meno>` musí byť jedno z následujúcich: *LimeWire, Bear-Share, Gnucleus, MorpheusOS, XoloX, MorpheusPE, gtk-gnutella, Acquisition, MyNapster, Mutella-0.4.1, MyNapster, Mutella-0.4, Qtella, AquaLime, Napshare, Comeback, Go, PHEX, SwapNut, Mutella-0.4.0, Shareaza, Mutella-0.3.9b, Morpheus, FreeWire, Openext, Mutella-0.3.3, Phex.*

Kvôli presnejšej detekcii sú do vyhľadávania zahrnuté reťazce `GET` a `HTTP`. Vďaka nim môžeme objaviť pakety, ktoré nezačínajú reťazcom `GNUTELLA` a nie sú typu `HTTP`.

[4] pre komunikáciu nevyžaduje žiaden server



Obrázek 2.3: Ukážka topológie siete aplikácie eMule (prevzaté z [21]).

2.3.2 eDonkey

Informácie ohľadom tohto protokolu sú čerpané z publikácie [21]. Charakteristiky protokolu eDonkey popíšem na najrozšírenejšej aplikácii využívajúcej tento protokol *eMule*. Tento typ siete pozostáva z niekoľko stoviek serverov a niekoľko miliónov klientov(viď Obrázok 2.3). Klient sa pripojí na niektorý zo serverov a zostáva naň pripojený počas celej doby, keď je aplikácia spustená. Servere poskytujú centralizované indexové služby a nekomunikujú so žiadnym iným serverom.

Na serveri sa nachádza databáza, v ktorej sú udržiavané informácie o klientoch a ich zdieľaných súboroch. Server neobsahuje súbory samotné. Na výmenu dát (zoznamy serverov, zoznam zdieľaných súborov, zoznamy iných klientov a iné) je využívané výhradne TCP spojenie. UDP spojenie sa využíva iba zriedka na signalizačné správy medzi klientmi alebo klientom a servom.

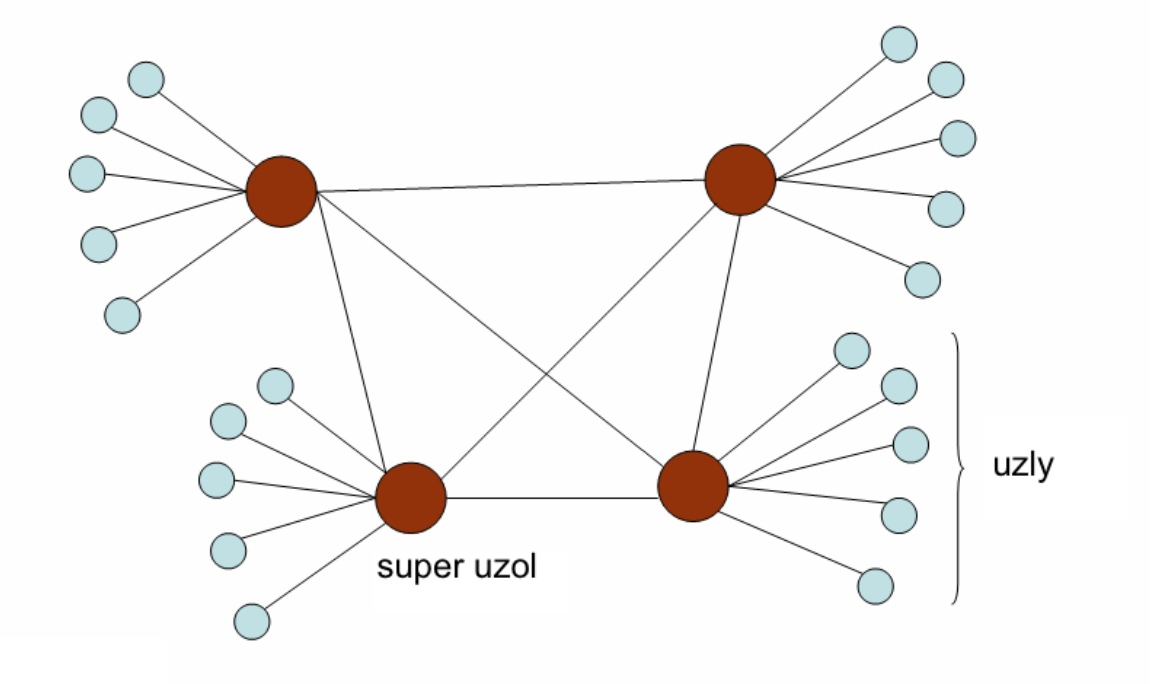
Štandardne má klient nadviazané spojenia s viacerými klientmi v sieti a navzájom si vymieňajú údaje, napr. súbory. Každý súbor je rozdelený do častí, tzv. *fragmentov*. Rozdelenie súborov do častí má veľký význam pri sťahovaní jedného súboru zároveň z viacerých zdrojov. Aplikácia sa zaregistruje naraz u viacerých klientov, že požaduje stiahnutie určitej časti súboru. Ak je daná časť súboru už stiahnutá a niektorý klient odpovie, že je možné od neho danú časť stiahnuť, táto odpoveď bude ignorovaná.

2.3.3 FastTrack

Informácie ohľadom tohto protokolu sú čerpané z publikácie [10].V sieti protokolu FastTrack vystupujú 2 základné prvky, ktoré medzi sebou komunikujú:

- **uzol** (anglicky *node*)
- **super uzol** (anglicky *super node*)

Super uzly slúžia ako centrálna databáza zoznamu súborov, pripojených užívateľov a na vyhľadávanie súborov. Klienti (uzly) sa pripájajú iba k super uzlom, ktoré im sprostredkúvajú všetky požadované informácie(viď Obrázok 2.4). Super uzly zároveň pôsobia v komunikácii ako spojovacie uzly, cez ktoré tečú všetky dáta. Vďaka tejto vlastnosti majú v porovnaní s obyčajnými uzlami oveľa vyššie nároky na výpočtový výkon počítača aj na šírku prenosového pásma. Uzly sa môžu meniť na super uzly a naopak. Podmienky, či sa z uzlu stane super uzol, sa odvíjajú od výpočtového výkonu počítača, na ktorom je táto aplikácia spustená a zároveň od šírky dostupného prenosového pásma.



Obrázek 2.4: Ukážka topológie siete protokolu FastTrack (podľa [10]).

2.3.4 BitTorrent

Sieť protokolu BitTorrent pozostáva z centralizovaného servera a klientov. Úloha servera spočíva v koordinovaní akcií klientov a správe spojení. Primárny účel, za ktorým bol tento protokol vyvinutý. je distribúcia objemných dát (napr. filmy, hudba a iné). Tvorcovia sa zamerali na efektívnosť protokolu. Dosiahli ju tým, že znížili šírku prenosového pásma i hardvérové nároky. Tieto pozitívne vlastnosti prispeli k veľkej obľúbenosti protokolu a neraz je využívaný komerčnými spoločnosťami, či na šírenie distribúcií operačného systému Linux.

Pre sťahovanie pomocou protokolu BitTorrent je potrebné získať súbor s príponou *torrent*. Zvyčajne je stiahnuteľný z webových stránok. Obsahom tohto súboru sú dáta, ktoré informujú klientskú aplikáciu o zdieľaných prostriedkoch (zoznam užívateľov, od ktorých je možné požadovaný súbor stiahnuť a iné). Následne sa aplikácia pripája k ostatným klientom a sťahuje od nich požadovaný súbor po častiach takým istým spôsobom ako protokol eDonkey popísaný v kapitole 2.3.2. Veľkosť častí, do ktorých je súbor rozdelený môže byť rôzna. Zvyčajne sa jedná o veľkosti 256kB, 512kB alebo 1MB. Ku každej z týchto častí je vytvorený kontrolný súčet pomocou SHA1 algoritmu [6].

Pri sťahovaní segmentov sa používa algoritmus, ktorý rozhoduje o priorite sťahovania jednotlivých častí súboru. Medzi prvými sú sťahované časti vyskytujúce sa na čo najmenšom počte klientov. Následne sa sťahujú časti vyskytujúce sa vo väčšom množstve. Na záver sa zisťuje, ktoré časti súboru ešte chýbajú a sú následne stiahnuté. Z tohto dôvodu môžme pri sťahovaní súboru pomocou protokolu BitTorrent zaznamenať 3 fázy sťahovania, od ktorých závisí rýchlosť získavania dát. Rýchlosť sťahovania dát je najvyššia v strednej fáze a najnižšia v počiatočnej a koncovej fáze.

2.4 Detekcia peer-to-peer sietí

Pre úspešné blokovanie/obmedzovanie peer-to-peer sietí je najdôležitejšou časťou ich detekcia. Žiadne peer-to-peer spojenie nemôže byť obmedzené bez predošlej úspešnej detekcie. Existuje viacero spôsobov, ktorými je možné odhaľovať tieto spojenia. Vzhľadom na neustále meniacu sa situáciu s vývinom peer-to-peer aplikácií, sa menia aj výhody a nevýhody jednotlivých spôsobov detekcie. Každá metóda sa snaží klásť veľký dôraz na samotnú detekciu peer-to-peer sietí, ale nesmie zabudnúť minimalizovať počet falošne identifikovaných spojení. Pokiaľ by bolo percento falošne identifikovaných spojení príliš veľké a následne by sme blokovali tieto spojenia, mohlo by dôjsť k zablokovaniu služieb, ktoré vykazujú podobné správanie ako peer-to-peer aplikácie (napr. DNS, skype a iné). Najpoužívanejšími spôsobmi detekcie sú:

- detekcia pomocou čísel portov
- detekcia dát na aplikačnej vrstve modelu ISO/OSI[5]
- Detekcia analýzou sieťovej premávky

2.4.1 Detekcia pomocou čísel portov

Jedná sa o najstaršiu a najjednoduchšiu formu detekcie/blokovania/obmedzovania peer-to-peer sietí. Metóda využíva znalosť štandardne používaných portov jednotlivých aplikácii (viď Tabuľka 2.4.1). Kedysi sa jednalo o najúčinnejší spôsob detekcie, no v súčasnosti nenachádza príliš veľké uplatnenie. Dôvodom je, že tieto porty sú v súčasnosti ľahko meniteľné. Aplikácie generujú čísla portov náhodne, alebo si ich môže užívateľ ľahko zmeniť.

Aplikácie sú taktiež schopné maskovať svoje porty. Princíp maskovania spočíva v tom, že peer-to-peer aplikácie sú schopné pre svoju komunikáciu vyžívať porty, ktoré sú pridelené iným aplikáciám (napr. port 80 pre HTTP, port 443 pre HTTPS, port 21 pre FTP a iné).

[5] bližšie informácie o modele ISO/OSI na [13]

Názov protokolu	Čísla portov
BitTorrent	6881-6999 [8]
Direct Connect	411-412
eDonkey2000	4662, 4672 [9]
Gnutella	6346-6347 [11]
Kazaa	1214 [11]

Tabulka 2.1: Prehľad známych portov niektorých peer-to-peer aplikácií.

2.4.2 Detekcia dát na aplikačnej vrstve modelu ISO/OSI

Tento typ detekcie sa zvykne označovať aj ako *detekcia analýzou payloadu*. Táto metóda má väčšiu účinnosť v detekcii peer-to-peer sietí ako predošlá. Narozdiel od metódy, v ktorej sme blokovali známe porty, sa v tejto metóde zameriame na analýzu dátovej časti paketu.

Každý peer-to-peer protokol v komunikácii posiela špecifické reťazce (viď Tabuľka 2.2), pomocou ktorých je možné tento protokol identifikovať. Tieto špecifické reťazce sa vyskytujú na začiatku dátovej časti paketu. Pre správnu identifikáciu takéhoto spojenia potrebujeme napísať pravidlá, s ktorými budú dáta porovnávané. Po úspešnej zhode s niektorým pravidlom, môžme prehlásiť, že daný paket patrí peer-to-peer aplikácii a podľa potrieb ho upraviť (zahodiť, spomaliť prenos a iné).

Názov protokolu	Špecifické reťazce		

BitTorrent	`GET /announce?info_bash`
	`GET /torrents/`
	`GET TrackPack`
	`0x13 BitTorrent`		

Gnutella	`GNUTELLA`
	`GET /uri-res/`
	`X-Gnutel`
	`GET /get/`		

Direct Connect	`$Send`
	`$Get`
	`$Key`
	`$Direction`
		`$Search`
		`$MyInfo`
		`$Hello`
		`$Lock`
			`$Connect`
			`$MyNick`
			`$Quit`
			`$Pin`

Fasttrack	`GET /.hash` `GIVE`		

Tabulka 2.2: Špecifické reťazce pre niektoré peer-to-peer protokoly.

Nevýhody [2]:

1. metóda nie je schopná automaticky sa prispôsobiť neznámym (novo-vzniknutým) protokolom
2. kontrola každého paketu vyžaduje veľký značný výkon
3. pomocou tejto metódy nie je možné identifikovať pakety, ktoré sú šifrované

2.4.3 Detekcia analýzou sieťovej premávky

Detekcia analýzou sieťovej premávky je z historického hľadiska najnovšou metódou na detekciu peer-to-peer sietí. Narozdiel od predošlej metódy, v ktorej sme analyzovali dáta obsiahnuté v pakete, sa snažíme analyzovať sieťové toky, ktoré jednotlivé peer-to-peer aplikácie vytvárajú (získavaním informácií z hlavičky paketu [3]).

Tento cieľ sa snažíme dosiahnuť sledovaním nasledujúcich údajov:

- typ protokolu (TCP/UDP)
- veľkosť paketu
- počet prijatých paketov za sekundu
- veľkosť sieťovej premávky
- pár IP adries (zdrojová-cieľová IP adresa)
- čísla portov
- počet úspešných a neúspešných spojení

Výhody oproti predchádzajúcim metódam:

1. odstránili sme problém so šifrovanými tokmi, pretože sa zameriavame na časť paketu, ktorá nie je šifrovaná
2. metóda nie je závislá na špecifických protokoloch - dokáže odhaliť aj novo vzniknuté protokoly
3. vďaka analyzovaniu inej ako dátovej časti paketu, odpadá problém s narušovaním súkromia užívateľov

Nevýhody:

1. veľmi náročné je odhadnúť správanie peer-to-peer aplikácií a nájsť vhodný algoritmus, ktorý toto správanie odhaľuje
2. falošne môžu byť označené aplikácie, ktoré majú podobné správanie ako peer-to-peer, no v skutočnosti peer-to-peer aplikáciami nie sú
3. správanie peer-to-peer aplikácií sa môže časom zmeniť, čo spôsobí, že náš algoritmus nedokáže tieto aplikácie odhaliť

Kapitola 3

Použité nástroje

Všetky nástroje, ktoré využívam vo svojej bakalárskej práci sú voľne dostupné (nie je potrebné ich registrovať, alebo platiť za ich registráciu). Sú licencované pod GNU GPL[1]/GNU GPLv2.

3.1 L7-filter

Informácie o tomto nástroji sú čerpané z publikácie [16]. L7-filter je linuxový nástroj slúžiaci na klasifikovanie, značenie paketov. Začal vznikať v roku 2003 ako odpoveď na veľmi drahé programy schopné kontrolovať šírku prenosového pásma, ktoré mali veľké problémy s prispôsobením na meniace sa protokoly. V máji 2003 bola uvoľnená prvá verzia tohto nástroja ako patch pre QoS[2] do Linuxového jadra. Zistilo sa, že verzia pre QoS nie je najvhodnejším riešením, tak sa vývoj zameral na Netfilter. Verzia pre Netfilter bola úspešne spustená v januári 2005. Nakoniec sa upustilo od implementácie L7-filtru do Linuxového jadra a vznikla verzia, ktorá beží v tzv. user space[3] a dáta získava pomocou QUEUE z Netfiltru.

Vďaka implementácii využívajúcej Netfilter, môže byť výsledok klasifikácie pomocou L7filtru použitý akýmkoľvek spôsobom, ktorý umožňuje nástroj Netfilter. L7-filter nedokáže pracovať s IPv6 ani ICMPv6.

Nástroj L7-filter porovnáva dáta na aplikačnej vrstve s regulárnymi výrazmi. Na základe zhody regulárneho výrazu rozlišuje špecifické protokoly. Dokáže odhaliť aj spojenia, ktoré:

- používajú nepredvídateľné porty (napr. zdieľanie súborov pomocou peer-to-peer)
- nepoužívajú štandardné porty (napr. HTTP[4] komunikácia bežiaca na porte 1111)
- zdieľajú rovnaké číslo portu (napr. zdieľanie súborov pomocou peer-to-peer používajúce port 80, ktorý je štandardne používaný protokolom HTTP)

Štandardne je porovnávaných prvých 10 paketov alebo 12 000 Bytov z každého spojenia (záleží od toho, ktorá z týchto 2 podmienok bude splnená skôr). Obe tieto nastavenia sa dajú prispôsobiť podľa potrieb. Výnimočne môže nastať, že spojenie môže byť zhodné so vzormi regulárnych výrazov (ďalej vzormi) viacerých protokolov. Vzory sú testované v presnom poradí, uvedenom v konfiguračnom súbore. Ak sa u daného toku nájde zhoda

[1] GNU GPL - *General Public License* je typ licencie, pri ktorej autor zdarma poskytuje kópie svojho programu. Akákoľvek časť takéhoto programu môže byť použitá v iných programoch [12]

[2] QoS - *Quality of Service* je schopnosť poskytovať rôznym dátovým tokom rôznu prioritu (viac na [1])

[3] user space - reprezentuje bežné užívateľské prostredie, v ktorom je možné spúšťať aplikácie

[4] HTTP - *HyperText Transfer Protocol* je protokol na prenos hypertextových dokumentov [12]

s niektorým vzorom, všetky pakety, patriace tomuto toku, sú označené značkou (definovanou v konfiguračnom súbore) a daný tok už nebude viac testovaný.

Vzory sú uložené v súboroch s príponou *pat* (napr. gnutella.pat). Všetky takto vytvorené súbory by sa mali nachádzať v jednej spoločnej zložke. Názov tejto zložky je predaný L7-filtru ako parameter pri jeho spustení. Obsahom týchto súborov je:

- názov protokolu (v našom prípade gnutella)
- regulárny výraz pomocou ktorého je možné daný tok odhaliť

Názov súboru (časť pred príponou) musí byť zhodná s názvom protokolu vo vnútri súboru. Pokiaľ tento názov nie je rovnaký, L7-filter sa nespustí a vypíše chybové hlásenie. Do jednotlivých vzorov je možné pridať komentáre pridaním znaku # pred tento komentár.

V konfiguračnom súbore aplikácie sa určuje, aké číslo značky bude priradené konkrétnemu vzoru. Pre akýkoľvek vzor je možné použiť značku s číslom 3 a vyššie. Značky 0-2 sú špecifické pre L7-filter, a preto by nemali byť použité pre žiaden vzor. Významy značiek 0-2:

- 0 - paket s touto značkou doposiaľ neprešiel L7-filtrom
- 1 - L7-filter nenašiel zhodu, pokúsi sa identifikovať nasledujúci paket daného spojenia
- 2 - nenašiel zhodu a daný tok už nebude identifikovať

Vo svojej bakalárskej práci využívam typ L7-filtru user space.

3.2 Iptables

Iptables je Linuxový nástroj, ktorý umožňuje nastaviť rôzne filtre pre jednotlivé pakety. Umožňuje prácu s IPv4 aj s IPv6. Pre jeho správnu funkcionalitu vyžaduje linuxové jadro s podporou paketového filtru *ip_tables*.

Iptables obsahuje 4 rôzne tabuľky, z ktorých každá obsahuje niekoľko typov reťazcov. Jeden reťazec (ďalej budem používať anglický výraz *chain*) predstavuje súbor pravidiel, pomocou ktorých je paket upravovaný. Paket prechádza postupne každým zadefinovaným pravidlom v chaine. Pokiaľ spĺňa podmienku akéhokoľvek pravidla z chainu, je naňho dané pravidlo aplikované. Po aplikovaní pravidla paket prechádza ďalšími pravidlami v chaine, ktoré nasledujú za práve aplikovaným pravidlom. Výnimkou je pravidlo, ktoré zahadzuje paket. Po dosiahnutí tohto pravidla paket ďalšími pravidlami neprechádza. Prechod paketu môžme vidieť na Obrázku 3.1. Niektoré základné operácie s paketmi sú:

- akceptovanie paketu
- zahodenie paketu
- skopírovanie paketu a jeho prípadné preposlanie na určitú IP adresu
- označenie paketu celočíselnou značkou
- zistenie značky paketu
- zistenie veľkosti dát prenášané paketom
- a iné

Tabuľky nástroja iptables a ich špecifické chainy [18]:

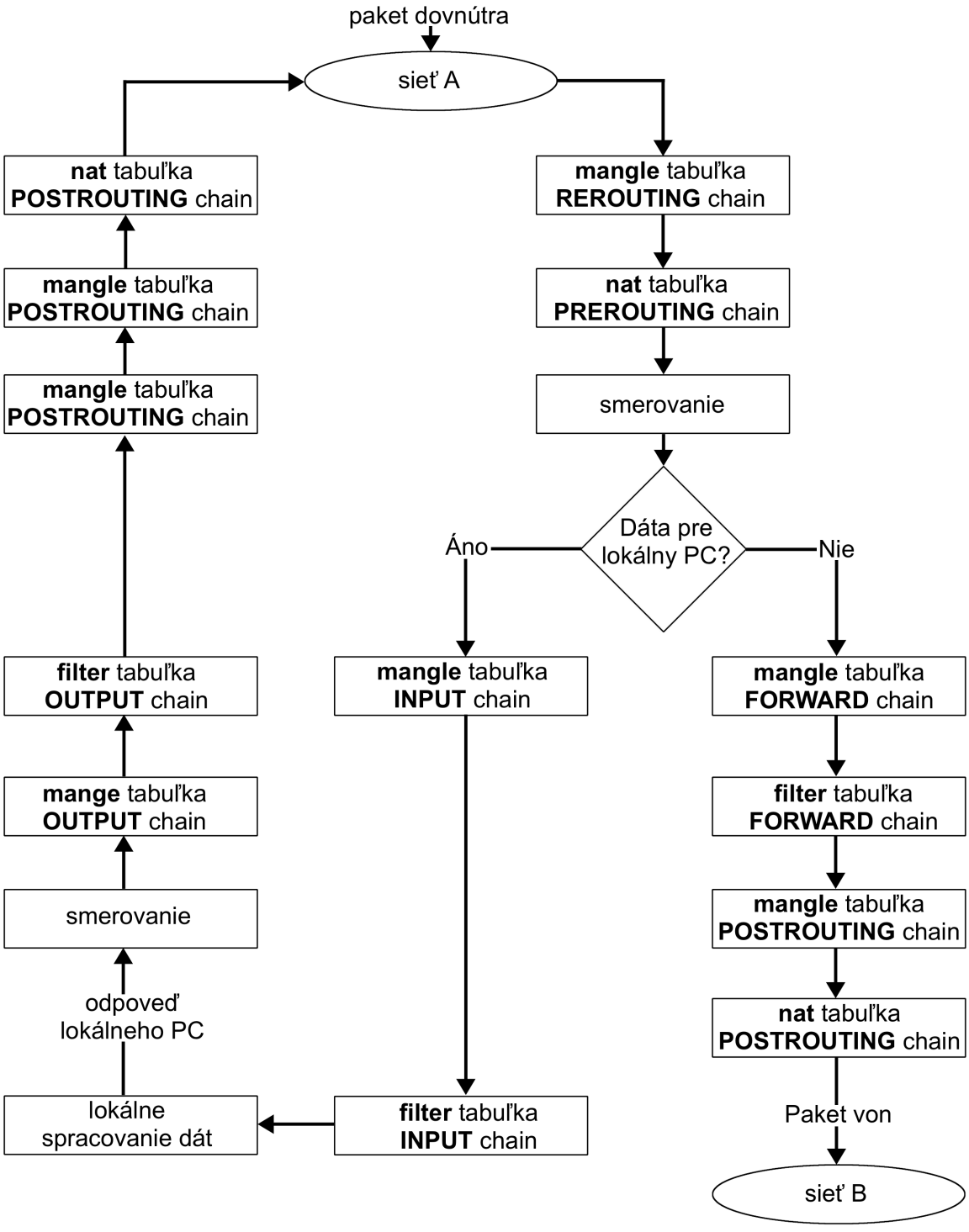
- **filter** - predvolená tabuľka, do ktorej sa zapíšu všetky pravidlá, pokiaľ v pravidle nie je striktne vyžiadaný zápis do inej tabuľky. Obsahuje vstavané chainy INPUT (pre pakety smerujúce na lokálne sockety), FORWARD (pre pakety určené na smerovanie), OUTPUT (pre pakety vygenerované lokálnym počítačom).
- **nat** - tabuľka využívaná, keď príde paket, ktorý vytvára nové spojenie. Obsahuje vstavané chainy PREROUTING (na vykonávanie zmien na paketoch hneď ako vstúpia do systému), OUTPUT (na vykonávanie zmien na paketoch pred smerovaním), POSTROUTING (na vykonanie posledných zmien na paketoch pred odoslaním).
- **mangle** - táto tabuľka sa používa na špecializovanú zmenu paketov. Do verzie linuxového jadra 2.4.17 obsahovala iba 2 vstavané chainy: PREROUTING (na vykonávanie zmien na paketoch pred smerovaním) a OUTPUT (na vykonávanie zmien na paketoch vygenerovaných lokálnym počítačom). Od verzie 2.4.18 boli pridané nasledujúce chainy: INPUT (pre pakety prichádzajúce na lokálny počítač), FORWARD(pre vykonanie zmien na paketoch smerované lokálnym počítačom) a POSTROUTING (na vykonanie posledných zmien na paketoch pred odoslaním).
- **raw** - táto tabuľka sa používa hlavne na nastavovanie výnimiek zo sledovaných spojení v kombinácii s cieľom NOTRACK. Zaregistruje sa nástroja netfilter s vyššou prioritou. vďaka čomu budú pravidlá z tejto tabuľky zavolané pred volaním ip_conntrack alebo inými IP tabuľkami. Obsahuje vstavané chainy: PREROUTING (pre pakety prichádzajúce cez ktorékoľvek sieťové rozhranie) a OUTPUT (pre pakety vygenerované lokálnymi procesmi)

Po aplikovaní ktoréhokoľvek pravidla si iptables zaznamenáva štatistiky o tom, koľko paketov spĺňalo príslušné pravidlo a aký objem dát bol daným pravidlom prenesený. Za pomoci týchto pravidiel je možné presmerovávať pakety do rôznych aplikácií, na iné IP adresy a zároveň umožňuje aj blokovanie paketov, ktoré spĺňajú príslušné pravidlo.

3.3 Traffic Control

Údaje o tomto nástroji sú čerpané z publikácie [20]. Traffic Control je Linuxový nástroj, pomocou ktorého môžme obmedzovať prenosové pásmo rôznych tokoch na úrovni paketov, zahadzovať pakety, označovať pakety a iné. Niektoré vlastnosti paketov, pomocou ktorých môžme zasahovať do prebiehajúcej komunikácie:

- značka paketu pridelená netfiltrom
- značka paketu pridelená traffic controlerom
- rozhranie, cez ktoré prichádzajú/odchádzajú pakety
- veľkosť paketov
- zdrojová/cieľová IP adresa
- zdrojový/cieľový port
- a iné



Obrázek 3.1: Ukážka prechodu paketu nástrojom iptables (podľa [17]).

Traffic Control pozostáva z jedinej fronty (typu FIFO[5]), do ktorej prichádzajú všetky pakety. Z fronty sa dostanú až vtedy, keď im to povolí hardvér.

Výhody:

1. pokiaľ je nástroj používaný správne, správanie siete je ľahšie predpovedateľné
2. pokiaľ nastavenie nástroja zodpovedá plánu dohodnutému s užívateľmi danej siete, užívatelia vedia presne, čo môžu od siete očakávať

Nevýhody:

1. nástroj je dosť náročný na pochopenie, pri jeho zlej konfigurácii je veľmi obtiažne nájsť príčinu problémov a následne ju odstrániť
2. nainštalovanie nesprávnym spôsobom vedie k veľmi vážnym problémom
3. pri použití na routeri musí mať router dostatočný výpočtový výkon, aby zvládal bez problémov činnosť tohto nástroja
4. aj keď v *packet-switched*[6] sieťach pokrýva väčšiu plochu, môžeme ho považovať za spôsob, ktorým poskytuje stavovosť *circuit-based* sietí *packet-switched*[7] sieťam

[5]FIFO - typ fronty, z ktorej odchádzajú prvky v takom poradí ako do nej vošli
[6]Bližšie informácie na [4]
[7]Bližšie informácie na [19]

Kapitola 4

Návrh systému

Na detekciu peer-to-peer sietí som si zvolil kombináciu detekcie pomocou známych portov a detekcie na úrovni aplikačnej vrstvy. Detekcia na úrovni aplikačnej vrstvy má oproti detekcii na základe sieťovej premávky výhodu, že dokážeme presne určiť, aký typ protokolu sa v našej sieti používa na peer-to-peer komunikáciu. Analýza pomocou tejto detekcie je rýchlejšia, nevyžaduje študovať presné správanie jednotlivých aplikácií (akým spôsobom vytvárajú jednotlivé spojenia a iné). Pri detekcii pomocou analýzy sieťovej premávky je nutné naprogramovať aplikáciu takým spôsobom, aby sa bola schopná učiť, kedy sa jedná o peer-to-peer spojenie a kedy nie. To vyžaduje veľkú časovú náročnosť i značný výkon počítača, na ktorom je takáto aplikácia umiestnená.

Z dôvodu nemožnej identifikácie šifrovaných spojení pri detekcii na úrovni aplikačnej vrstvy, je táto metóda rozšírená o možnosť blokovať špecifické porty.

4.1 Návrh užívateľského rozhrania

Pri návrhu užívateľského rozhrania som sa snažil zamerať na jeho jednoduchosť, prehľadnosť, intuitívnosť, rýchle pochopenie práce s ním. Užívateľovi by sme mali poskytnúť čo najľahší prístup k dátam, ktoré potrebuje zo systému získať. Dáta by mali byť prezentované pomocou grafov, ktoré budú umiestnené na záložkách, medzi ktorými sa bude dať prepínať. Tieto grafy bude možné pridávať po kliknutí na tlačítko, viditeľné na obrazovke, kde si môže užívateľ zvoliť, aké dáta potrebuje mať v grafe zobrazené. Po kliknutí na tlačítko *pridať*, bude do hlavného okna aplikácie pridaná nová záložka s požadovaným grafom. Ak by sa užívateľ pomýlil pri zadávaní údajov, ktoré chce mať vykreslené v grafe, môže tieto údaje upraviť po kliknutí na záložku s požadovaným grafom - na pravej strane od grafu sú zobrazené dáta, ktoré daný graf zobrazuje a v prípade potreby je možné tieto dáta zmeniť.

V hlavnom okne aplikácie môžme nájsť tlačítko na vyvolanie dialógu s nastaveniami aplikácie, v ktorých sa dá nastaviť:

- pripojenie do databáze (adresa servera, číslo portu, prihlasovacie meno, prihlasovacie heslo, názov databáze)
- či sa majú ukladať nové dáta do databáze
- cestu k programom *iptables, l7-filter, traffic control*
- časový interval, v akom sa majú získavať dáta z nástroja *iptables*

- protokoly, ktoré chce užívateľ sledovať (názov protokolu, cestu k súboru so vzorom, porty pridelené danému protokolu)
- IP adresy/podsiete, pre ktoré chce užívateľ získavať štatistiky
- obmedzenie rýchlosti pre jednotlivé sledované protokoly

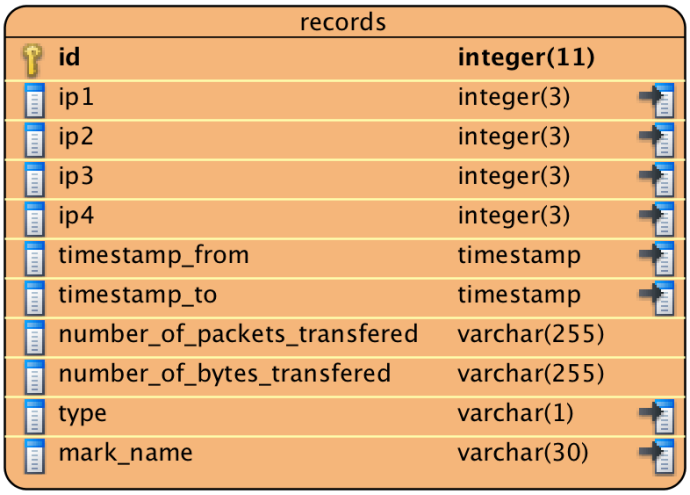
4.2 Návrh databáze

Nové získané dáta budú môcť byť na požiadanie ukladané do databázy. Budú do nej uložené iba v prípade, že je správne nastavené pripojenie do databázy a užívateľ si zapne ukladanie dát do databázy.

Dáta, ktoré si potrebujeme uchovávať:

- sledovaná IP adresa
- časový interval, v ktorom boli dáta zaznamenávané
- veľkosť dát prenesených za daný časový interval
- počet paketov prenesených za daný časový interval
- rozlíšenie typu záznamu - či sa týka prichádzajúcej alebo odchádzajúcej sieťovej premávky danej IP adresy
- hodnotu značky pre daný záznam
- názov protokolu pre každú značku

Štruktúra databáze viď Obrázok 4.1.



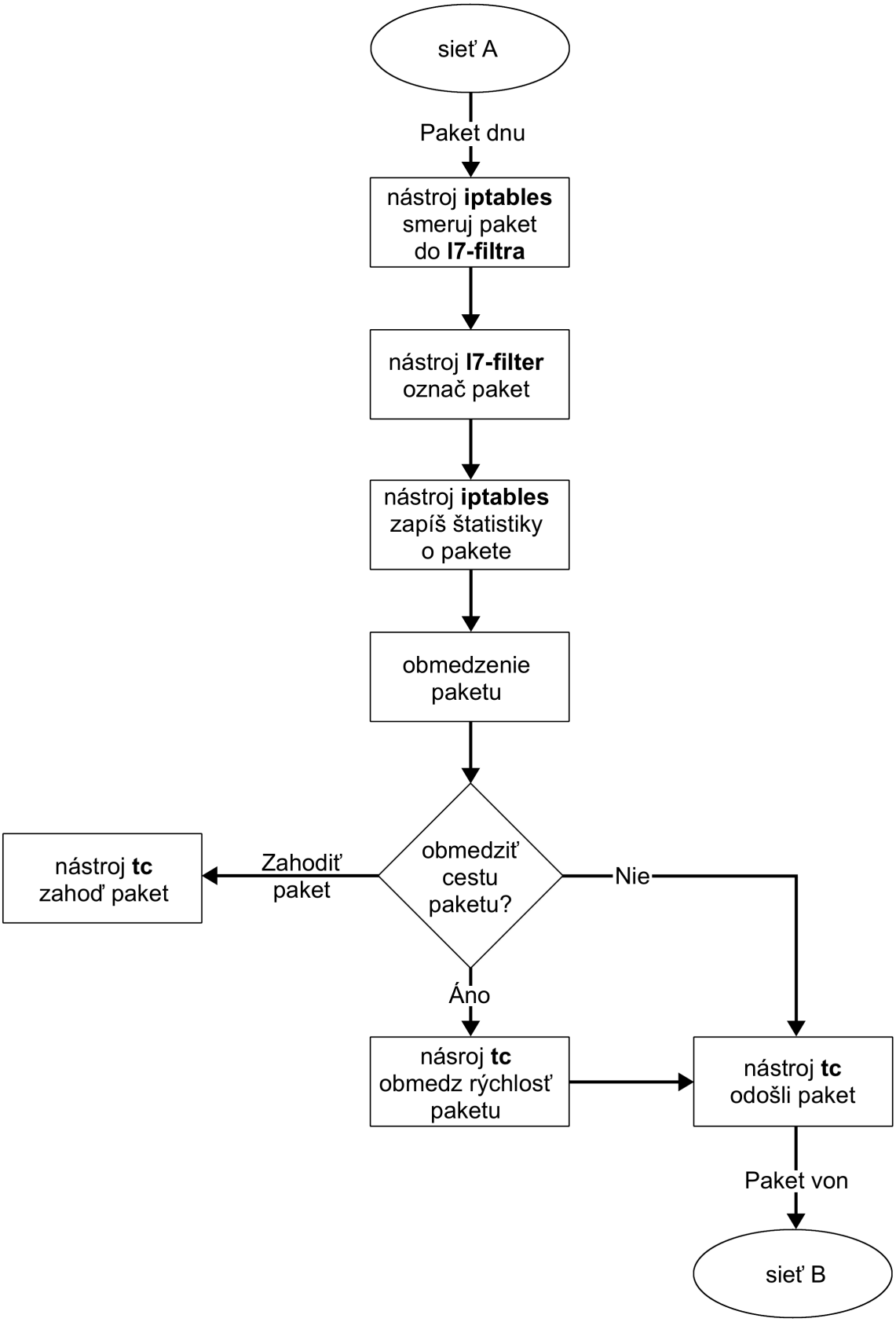
Obrázek 4.1: Návrh databáze.

4.3 Návrh prechodu paketu systémom

Cesta paketu mnou navrhovaným systémom je ovplyvnená viacerými nástrojmi, ktoré s týmto paketom pracujú. Keďže sa zameriavame na aplikáciu, ktorá by mala byť spustená na serveri (spojovací bod viacerých sietí/podsietí), zameriame sa iba na detekciu paketov smerované týmto počítačom. Pakety smerované na tento počítač, alebo generované týmto počítačom, nebudeme brať do úvahy. Cesta paketu systémom (viď Obrázok 4.2):

1. nástroj iptables vloží paket, ktorý má byť serverom presmerovaný, na vstup programu l7-filter
2. nástroj l7-filter sa pokúsi zistiť, či paket patrí spojeniu už zistenému, alebo spojeniu doposiaľ nezistenému. Pokiaľ tento paket patrí spojeniu, ktoré už bolo nejakým spôsobom zistené, priradí mu značku, ktorou bolo toto spojenie označené. V opačnom prípade sa pokúsi zistiť o aký typ spojenia sa jedná a podľa toho mu priradí patričnú značku.
3. nástroj iptables si zapíše štatistické informácie o pakete
4. nástroj traffic control na základe značky paketu a pravidiel odošle paket bez akéhokoľvek zásahu, obmedzí prenosovú rýchlosť paketu, alebo paket zahodí

Pokiaľ paket prejde celým systémom úspešne, je odoslaný na cieľovú adresu.



Obrázek 4.2: Prechod paketu systémom.

Kapitola 5

Implementácia

Ako cieľový operačný systém som zvolil Linux hlavne kvôli veľkému počtu voľne dostupných utilít, ktoré mi pomáhajú riešiť daný problém. Každá distribúcia Linuxu má v sebe zahrnutú väčšinu mnou používaných nástrojov. Pokiaľ niektorý z nástrojov nie je štandardnou súčasťou operačného systému, dá sa ľahko doplniť. Konkrétna mnou zvolená distribúcia Linuxu je: **Ubuntu 9.10 32-bit**.

Cieľový jazyk, v ktorom je napísaná bakalárska práca je **Java**. Tento jazyk som si vybral, pretože mám s ním najväčšie skúsenosti. Aplikácie napísané v tomto jazyku môžu byť ľahko prenesené do iného operačného systému, prípadne umiestnené na web. Pokiaľ by sme chceli preniesť moju aplikáciu na iný operačný systém, do ktorého nedokážeme implementovať mnou využívané nástroje, stačí nájsť vhodné nástroje v cieľovom operačnom systéme a re-implementovať triedy pracujúce s týmito aplikáciami.

5.1 Spôsob detekcie peer-to-peer spojení

Peer-to-peer siete som sa rozhodol detekovať pomocou nástroja L7-filter. V nastaveniach mojej aplikácie môžeme nastaviť potrebné atribúty, aby mohol tento nástroj pracovať správne. Potrebné nastavenie atribútov pre správne fungovanie aplikácie:

- správne nastavenie názvov protokolov a priradenie správneho súboru so vzorom špecifickému protokolu
- pokiaľ nie je l7-filter prístupný z ktoréhokoľvek miesta systému, je potrebné nastaviť cestu k tomuto nástroju
- je nutné dodržať konvencie pre písanie vzorov pre L7-filter podľa pravidiel napísaných na [16]

Každý zo súborov, v ktorých sú uložené vzory pre jednotlivé protokoly, sa môže nachádzať v akejkoľvek lokácii na disku. Pri spustení aplikácie alebo po uložení nastavení sú vykonané následovné operácie s aplikáciou L7-filter:

1. všetky súbory so vzormi, ktoré sú nastavené v aplikácii, sú skopírované do jednej zložky v adresárovej štruktúre aplikácie
2. vygeneruje sa konfiguračný súbor s nastaveniami pre špecifické protokoly
3. vypnú sa všetky spustené inštancie aplikácie L7-filter

4. spustí sa nová inštancia aplikácie L7-filter, kde sa jej v parametroch predá cesta ku konfiguračnému súboru a k zložke, v ktorej sú skopírované všetky súbory so vzormi

Po spustení novej inštancie aplikácia dokáže pomocou vzorov detekovať jednotlivé protokoly. Pakety sú označené príslušnou značkou, ktorá je definovaná v konfiguračnom súbore.

V prípade potreby je možné priradiť aplikáciám špecifické porty, na ktorých zvyknú komunikovať.

5.2 Spôsob získavania štatistík

Na získavanie štatistík využívam nástroj iptables (viac o nástroji v kapitole 3.2). Po spustení aplikácie alebo zmene nastavení sa vygenerujú pravidlá, pomocou ktorých je možné sledovať rôzne typy štatistík. Všetky vytvorené pravidlá vkladám do tabuľky *mangle* a chainu *POSTROUTING*.

Pravidlá sú generované pre každú IP adresu z nastavení alebo pre každú IP adresu zo zadanej podsiete (vrátane adresy siete). Pre každú z týchto IP adries je vygenerovaných niekoľko záznamov. Ich počet závisí od počtu sledovaných protokolov. Pre každý z týchto protokolov sú vygenerované 2 záznamy, z ktorých prvý reprezentuje prichádzajúcu sieťovú premávku na danú IP adresu a druhý reprezentuje odchádzajúcu sieťovú premávku z danej IP adresy. Následne sú do iptables vložené záznamy pre 2 špeciálne značky. Jedná sa o špeciálne značky používané L7-filtrom (viď kapitola 3.1). Z toho vyplýva, že pokiaľ sledujeme 5 protokolov, tak je do iptables pre 1 IP adresu vložených $2 \times 5 + 2 \times 2 = 14$ záznamov.

Príkaz pre vloženie záznamu pre 1 IP adresu(192.168.181.5) a protokol so značkou 3:

```
iptables -A POSTROUTING -t mangle -s 192.168.181.5 -m mark --mark 3
iptables -A POSTROUTING -t mangle -d 192.168.181.5 -m mark --mark 3
```

Iptables si ukladá štatistiky pre každé, takto vytvorené, pravidlo. Tieto štatistiky obsahujú 2 základné údaje:

- počet prenesených bajtov/kilobajtov/megabajtov/gigabajtov. ktoré spĺňali dané pravidlo
- počet paketov, ktoré spĺňali dané pravidlo

Tieto štatistiky sa dajú z iptables získať pomocou následujúceho príkazu:

```
iptables -L -t mangle -v -n -Z
```

Podrobnejšie informácie pre popis jednotlivých parametrov viď [14]. Pre zefektívnenie štatistík vrátených nástrojom iptables modifikujem tento príkaz pomocou regulárneho výrazu, vďaka ktorému mi iptables vráti štatistiky iba tých pravidiel, ktorých štatistiky majú nenulovú hodnotu.

Konečný príkaz vypadá následovne:

```
iptables -L -t mangle -v -n -Z | grep ^[\ A-Z]*[1-9a-zA-Z]
```

Tieto štatistiky sa získavajú z iptables pravidelne v intervale zadanom užívateľom. Po získaní sú spracované aplikáciou.

5.3 Spôsob obmedzovania rýchlosti spojení

Prenosovú rýchlosť obmedzujem pomocou nástroja traffic control (viac o nástroji v kapitole 3.3). Po spustení aplikácie alebo zmene nastavení sa vygenerujú pravidlá, pomocou ktorých je obmedzovaná sieťová premávka. Počet pravidiel záleží od počtu portov pridelených protokolu, ktorý chceme obmedziť a od počtu pravidiel. Ak máme vytvorené 1 pravidlo a blokovaný protokol má 4 pridelené porty, vygeneruje sa nám $1 + 4 = 5$ pravidiel.

Predstavme si, že máme protokol so značkou 3, ktorý má pridelené 2 čísla portov (44100, 44101). My chceme obmedziť všetku premávku tohto protokolu prichádzajúcu do našej podsiete (192.168.180.0/24) na 1 MB/s a na špecifickú adresu (192.168.180.5) na 256 kbit/s. Rozhranie, na na ktoré je pripojená privátna podsieť, má názov eth2. Pravidlá pre tento prípad budú vypadať následovne (viď Obrázok 5.1):

Zmazanie všetkých predošlých záznamov:

```
tc qdisc del dev eth2 root
```

Vloženie hlavnej triedy pre rozhranie eth2:

```
tc qdisc add dev eth2 root handle 2:0 cbq avpkt 1000 bandwidth 100mbit
```

Vloženie podtried pre obe ip adresy, kde im nastavíme prenosové rýchlosti:

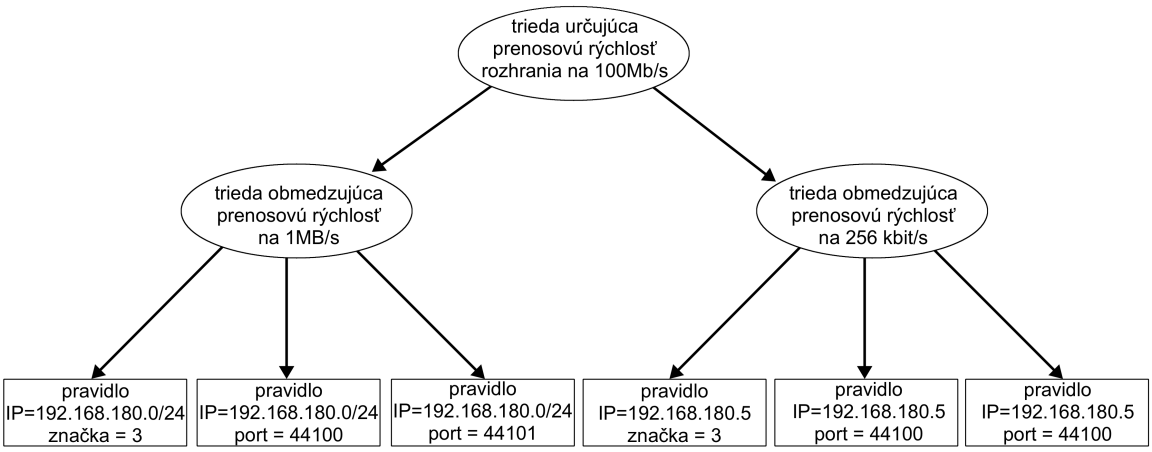
```
tc class add dev eth2 parent 2:0 classid 2:1 cbq rate 1mbps \allot 1500 prio 5
bounded isolated
tc class add dev eth2 parent 2:0 classid 2:2 cbq rate 256kbit \allot 1500 prio 5
bounded isolated
```

Vloženie filtrov pre jednotlivé podtriedy:

```
tc filter add dev eth2 protocol ip parent 2:0 u32 \match mark 3 0x3
 \match ip dst 192.168.181.0/24 flowid 2:1
tc filter add dev eth2 protocol ip parent 2:0 u32 \match ip dport 44100 0xffff
\match ip dst 192.168.181.0/24 flowid 2:1
tc filter add dev eth2 protocol ip parent 2:0 u32 \match ip dport 44101 0xffff
\match ip dst 192.168.181.0/24 flowid 2:1
tc filter add dev eth2 protocol ip parent 2:0 u32 \match mark 3 0x3
\match ip dst 192.168.181.5 flowid 2:2
tc filter add dev eth2 protocol ip parent 2:0 u32 \match ip dport 44100 0xffff
\match ip dst 192.168.181.5 flowid 2:2
tc filter add dev eth2 protocol ip parent 2:0 u32 \match ip dport 44101 0xffff
\match ip dst 192.168.181.5 flowid 2:2
```

Pravidlá s označením *flowid 1:1* zdieľajú prenosovú triedy s označením it 1:1 1 MB/s. Do tejto triedy patria následovné 3 pravidlá:

1. značka je rovná 3 a zároveň cieľová IP adresa je 192.168.181.0/24
2. cieľová IP adresa je 192.168.181.0/24 a zároveň cieľový port je 44100
3. cieľová IP adresa je 192.168.181.0/24 a zároveň cieľový port je 44101



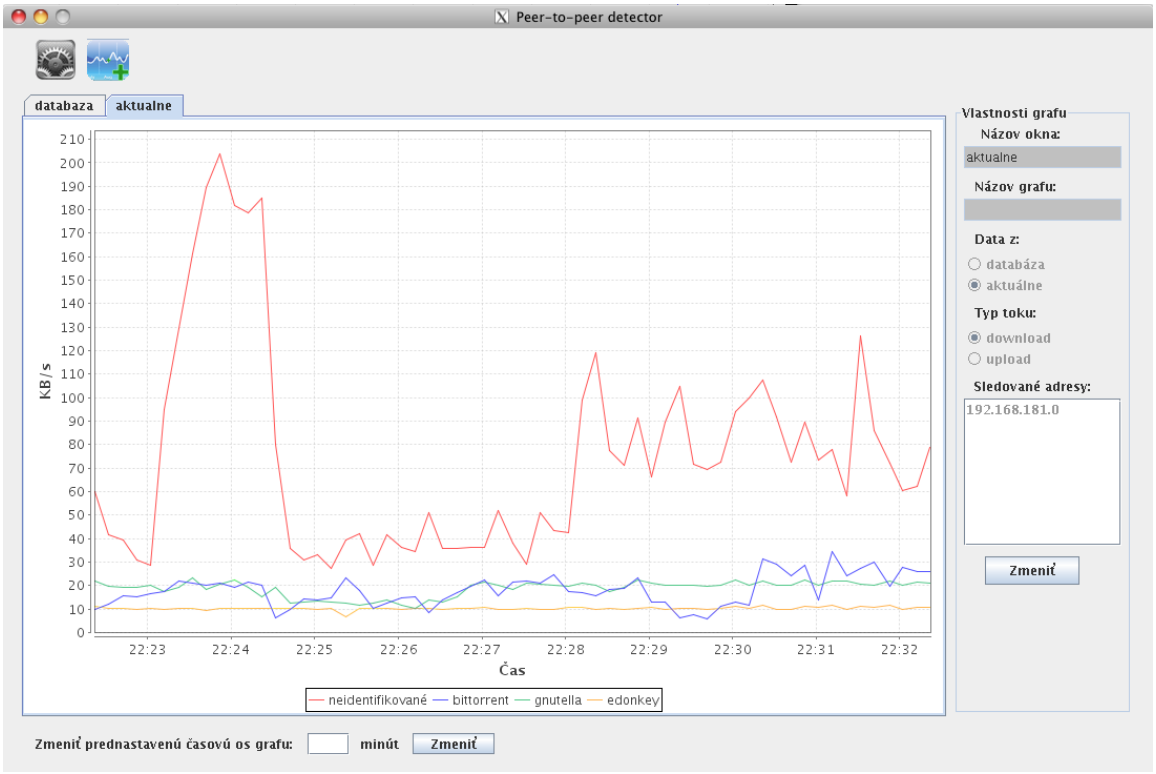
Obrázek 5.1: Grafická prezentácia pravidiel z kapitoly 5.3.

5.4 Spôsob prezentovania získaných dát

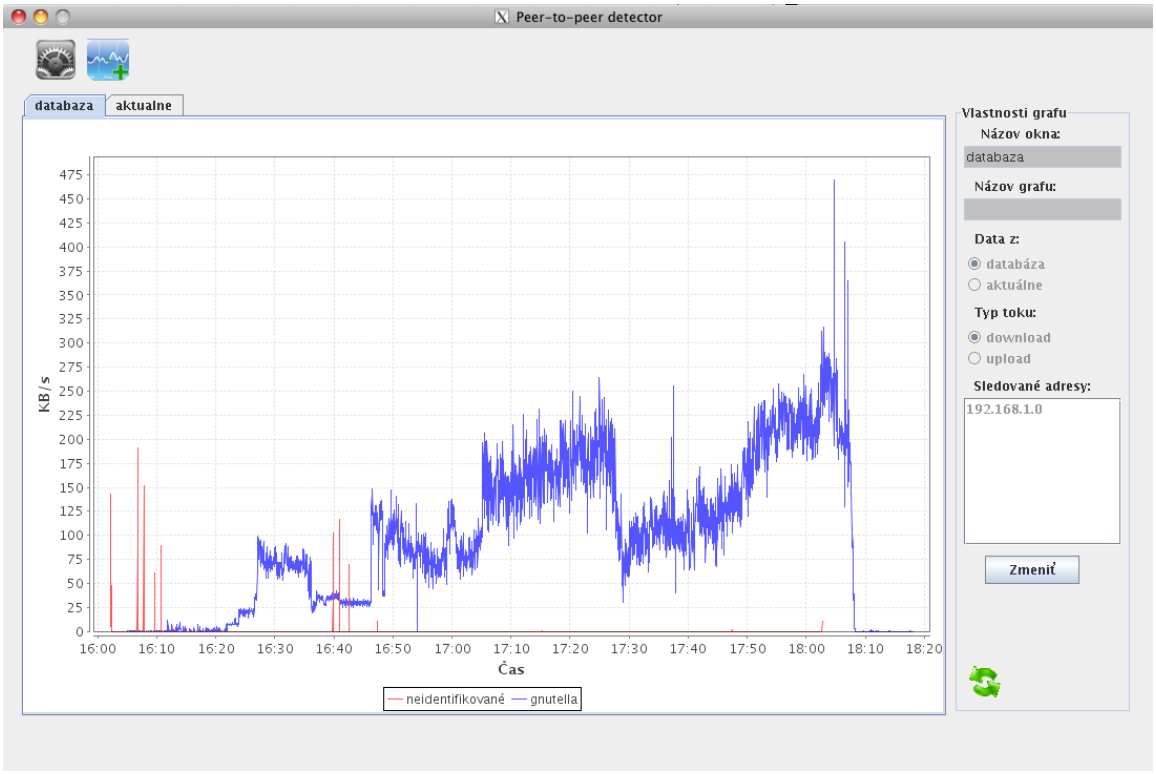
Dáta v aplikácii reprezentujem pomocou grafov. Na prácu s grafmi využívam voľne dostupnú knižnicu *JFreeChart* (viac o knižnici na [15]). Jedná sa o voľne dostupnú knižnicu.

S dátami získané pomocou iptables môžeme vykonávať viacero úkonov:

- môžeme si nechať vykreslovať graf pre danú IP adresu alebo podsieť. U nich si zvolíme, či chceme sledovať prichádzajúcu alebo odchádzajúcu sieťovú premávku (viď Obrázok 5.2)
- nastavíme, aby sa všetky získané dáta ukladali do databázy
- necháme si z databáze vypísať zachytené dáta pre danú IP adresu alebo podsieť, u ktorej si zvolíme, či chceme vidieť prichádzajúcu alebo odchádzajúcu sieťovú premávku (viď Obrázok 5.3)



Obrázek 5.2: Ukážka grafu, ktorý zobrazuje aktuálne dáta.



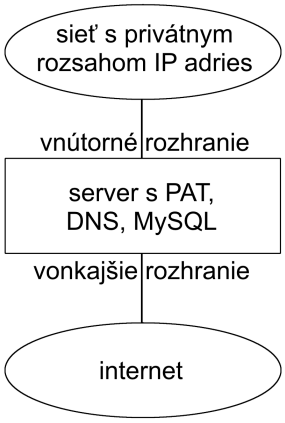
Obrázek 5.3: Ukážka grafu, ktorý má načítané dáta z databáze.

Kapitola 6

Testovanie

Nástroj bol testovaný na počítači s parametrami: AMD Athlon(tm) 64 X2 Dual Core Processor 6000+, 2 GB RAM. Fungoval ako plnohodnotný smerovač, na ktorom bol spustený preklad ip adries (PAT), mysql databáza a preklad doménových mien (DNS). Na vnútornom sieťovom rozhraní sa nachádzala sieť v rozsahu privátnych IP adries a na vonkajšom rozhraní sa nachádzalo pripojenie k sieti internet (viď Obrázok 6.1). Do vnútornej siete bol pripojený počítač s operačným systémom Windows XP, na ktorom boli nainštalované rôzne peer-to-peer programy, ktorých komunikáciu som na serveri analyzoval. Každým programom bol pri teste sťahovaný súbor s veľkosťou približne 700 MB.

Všetky nástroje boli testované so štandardnými nastaveniami, s akými boli stiahnuté z internetových zdrojov.

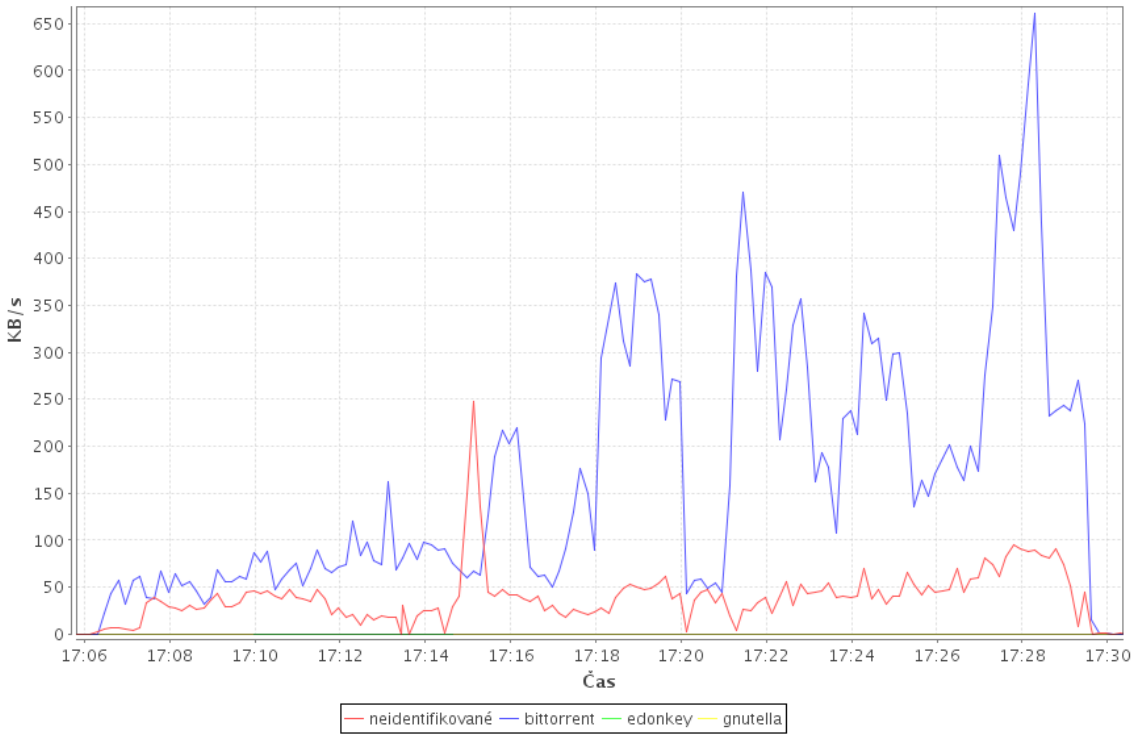


Obrázek 6.1: Ukážka zapojenia serveru v sieti.

6.1 Detekcia protokolu BitTorrent

Pomocou aplikácie uTorrent som sa pripájal do siete BitTorrent. Na internete som si stiahol súbor s príponou *torrent*, pomocou ktorého je možné nami požadované dáta získať. Celková dĺžka sťahovania tohto súboru trvala necelých 25 minút, pričom sťahovanie bolo ukončené dosiahnutím nastaveného limitu pre odosielané dáta na 700 MB (priebeh sťahovania viď Obrázok 6.2). Počas sťahovania klient hľadá nových klientov, od ktorých môže sťahovať požadované dáta. Zaznamenal som čiastočnú neidentifikovanú prichádzajúcu aj odchádzaj-

úcu sieťovú komunikáciu. Obsah tejto komunikácie sa nepodarilo identifikovať. S určitosťou však môžem povedať, že sa jedná o komunikáciu v sieti BitTorrent, pretože počas tohto testu na danom počítači neprebiehala žiadna iná sieťová komunikácia.



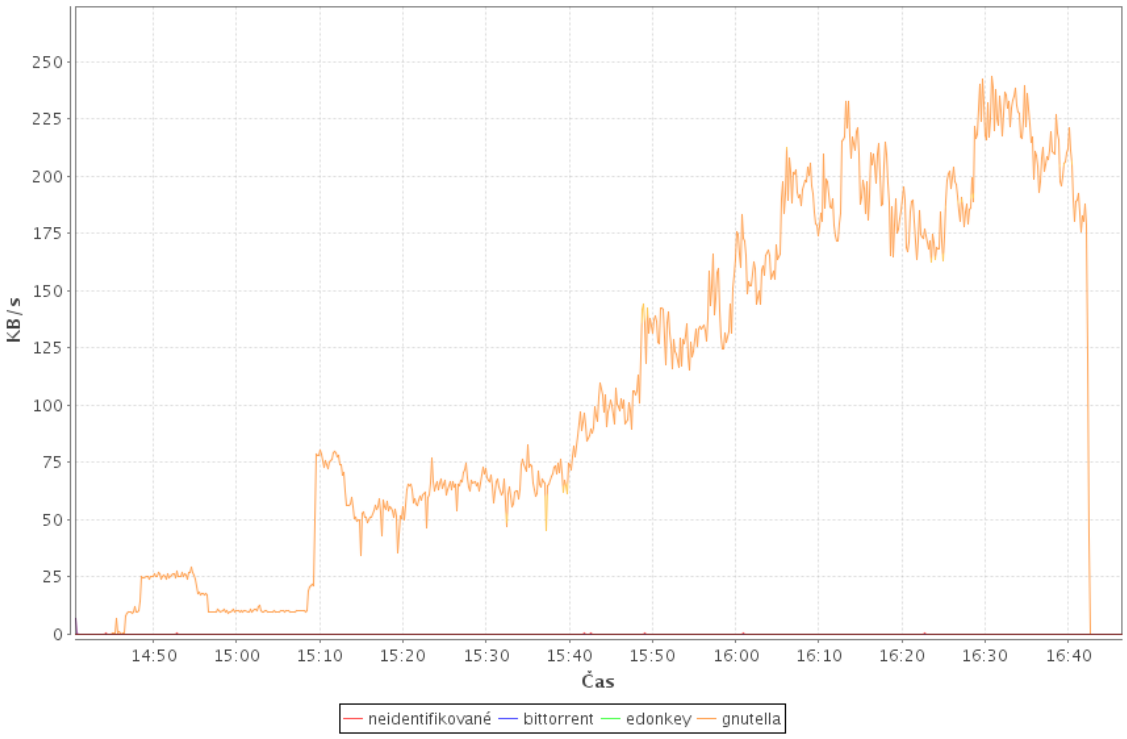
Obrázek 6.2: Prichádzajúca sieťová komunikácia protokolu BitTorent

6.2 Detekcia protokolu Gnutella

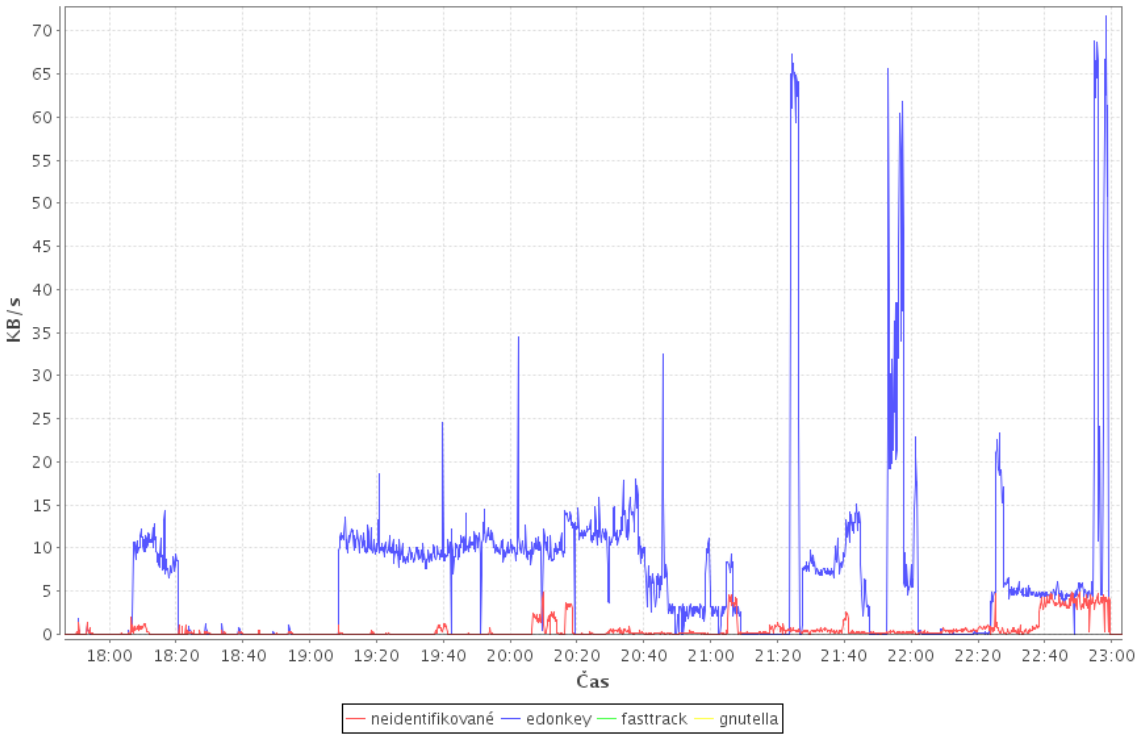
Do siete Gnutella som sa pripájal pomocou aplikácie Gnutella Turbo. Program sa najskôr pripája do siete, čo môže trvať aj niekoľko minút. Po úspešnom pripojení je možné vyhľadávať nami požadované súbory a následne ich sťahovať. Sťahovanie súborov neraz trvá veľmi dlho. Čas sa odvíja od počtu užívateľov, u ktorých sú dáta dostupné. Na obrázku 6.3 môžeme vidieť, že zo začiatku sťahovania nie je rýchlosť sťahovania veľmi vysoká, ale časom narastá. Prichádzajúca aj odchádzajúca sieťová premávka bola veľmi dobre detekovaná. Vďaka tomu, že som nezdieľal žiadne dáta, prenosové pásmo odchádzajúcej premávky ani raz nepresiahlo 5KB/s.

6.3 Detekcia siete eDonkey

Do siete typu eDonkey som sa pripájal pomocou aplikácie eMule. Pripojenie do siete tejto aplikácie bolo najkratšie zo všetkých testovaných aplikácií. Takmer ihneď bolo možné vyhľadávať nami požadované súbory. Veľký problém v tejto sieti je nájsť vhodné dáta na sťahovanie, pretože, ako môžme vidieť na obrázku 6.4, prenosová rýchlosť súborov nie je príliš vysoká. Je závislá na počte klientov, ktorí zdieľajú nami požadované dáta.



Obrázek 6.3: Prichádzajúca sieťová komunikácia protokolu Gnutella



Obrázek 6.4: Prichádzajúca sieťová komunikácia protokolu eDonkey

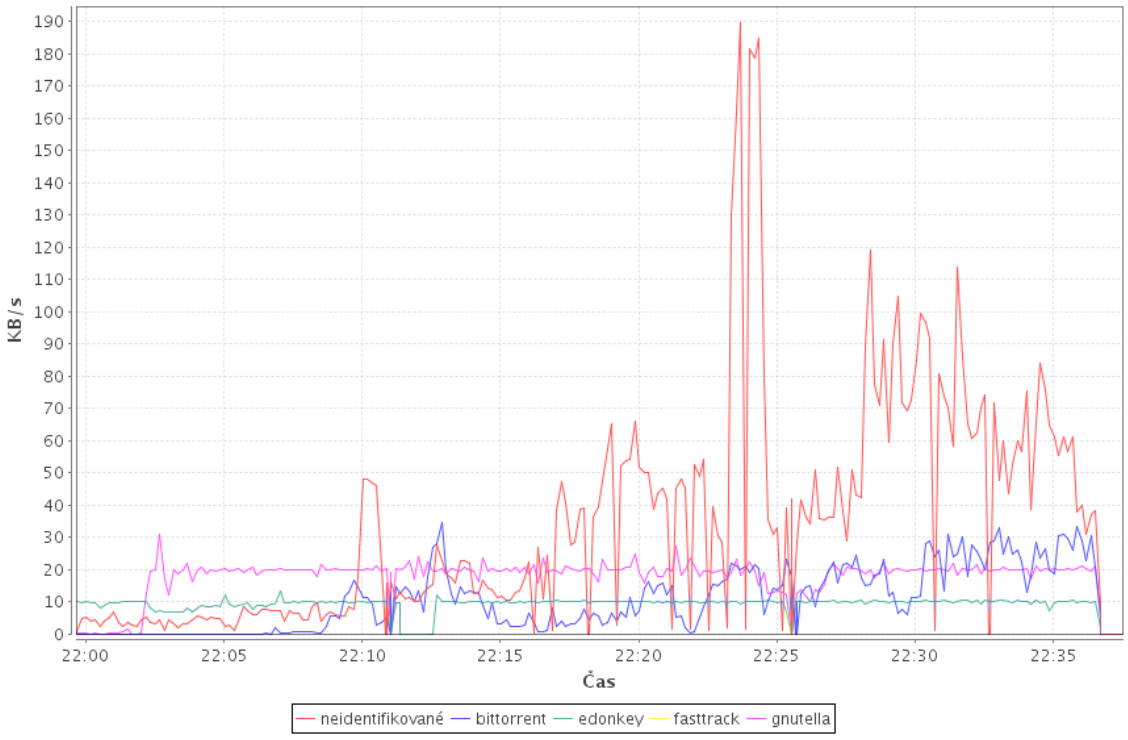
I po úspešnom nájdení väčšieho počtu klientov som musel veľmi dlho čakať, kým sa dáta začali sťahovať a rýchlosť bola malá. Pri ponechaní štandardných nastavení, má aplikácia eMule veľmi vysokú odchádzajúcu sieťovú premávku. Protokol je veľmi ľahko detekovaný, a preto sa dá ľahko obmedziť.

6.4 Testovanie obmedzenia peer-to-peer spojení

Po úspešnej detekcii peer-to-peer sietí som testoval, či je nástroj *traffic control* schopný na základe správnych pravidiel obmedziť toky týchto sietí na požadované hodnoty. Vytvoril som pravidlá na obmedzenie prichádzajúcej aj odchádzajúcej sieťovej premávky. Každý protokol mal nastavené limity separátne na prichádzajúcu aj odchádzajúcu sieťovú premávku na následujúce hodnoty:

- protokol bittorrent na 30 KB/s
- protokol gnutella na 20 KB/s
- protokol edonkey na 10 KB/s

Na obrázku 6.5 môžme vidieť, že detekovanej sieťovej komunikácii bola prenosová rýchlosť úspešne zredukovaná. V grafe si môžme taktiež všimnúť výkyvy nad povolenú hranicu. Toto je spôsobené správaním nástroja *traffic control*, ktorý umožňuje preniesť za krátky čas väčšie množstvo dát, než povoľuje nami napísané pravidlo a následne pozdrží pakety po takú dobu, aby bol limit dodržaný.



Obrázek 6.5: Ukážka obmedzenia rýchlostí peer-to-peer aplikácií

Kapitola 7

Záver

V tejto práci bola navrhnutá a implementovaná aplikácia schopná nájsť peer-to-peer siete a obmedziť ich prenosové pásmo. Za týmto účelom boli použité voľne dostupné linuxové nástroje umožňujúce nájsť peer-to-peer siete na úrovni aplikačnej vrstvy, označiť takéto toky, získať štatistiky o daných tokoch a zároveň tieto toky obmedzovať. Činnosť aplikácie bola otestovaná v ostrej sieťovej premávke, kde bola spustená na serveri, ktorý plní úlohou smerovača z privátnej siete do internetu. Vďaka činnosti našej aplikácie sme mali prehľad o protokoloch využívaných v našej privátnej sieti.

Nedostatkom aplikácie je, že nedokáže odhaľovať spojenia využívajúce šifrovaný prenos. Taktiež vzory využívajúce pri detekcii sa musia udržiavať aktuálne, aby mohla byť detekcia postačujúca. Vďaka umiestneniu na serveri, na ktorom je spustená služba PAT, nie je možné obmedzovať odchádzajúcu sieťovú premávku zo špecifických IP adries - je možné obmedziť iba celkovú sieťovú premávku špecifickú pre konkrétny protokol odchádzajúcu rozhraním, ktoré je pripojené do internetu.

Vhodným vylepšením do budúcna by mohlo byť pridanie niektorého z iných spôsobov detekcie (napr. detekciu analýzou sieťovej premávky). Integrovaním tohto spôsobu prispeje k lepšej detekcii a tým pádom pomôže poskytovateľom internetu lepšie spravovať svoje siete. Vhodným vylepšením by bola integrácia neurónových sietí do programu za účelom inteligentného rozpoznávania peer-to-peer sietí, učenie sa vlastností peer-to-peer sietí a iné.

Aplikácia je vďaka používaniu vzorov l7-filtru ľahko rozšíriteľná na prípadnú detekciu aj iných protokolov (protokolov, ktoré nie sú peer-to-peer). Stačí v programe pridať vzor, ktorý identifikuje takéto spojenia.

Testovaním a prácou s jednotlivými peer-to-peer programami som zistil, že ich správanie je neraz nepredvídateľné a navrhnutie správnych algoritmov na ich detekciu je nesmierne náročné.

I keď je detekcia peer-to-peer sietí a ich obmedzovanie dôležité, poskytovatelia internetu by sa nemali utiekať k úplnému odstraňovaniu tejto komunikácie zo svojich sietí. Mali by sa zamerať na redukovanie prenosových pásiem týchto aplikácií na prijateľnú hodnotu ako aj pre ISP, tak aj pre zákazníka. Pokiaľ sa nebudú poskytovatelia snažiť žiť v symbióze s týmito sieťami, bude kladený veľký dôraz na maskovanie peer-to-peer sietí a ich detekcia bude neustále náročnejšia.

Literatura

[1] A. Campbell, G. Coulson, D. Hutchison: *A quality of service architecture*. ACM New York, NY, USA, 2004, 6–27 s., iSSN 0146-4833.

[2] I. Dedinski, H. Meer, L. Han, L. Mathy, D. P. Pezaros, J. S. Sventek, X. Y. Zhan: Cross-Layer Peer-to-Peer Traffic Identification and Optimization Based on Active Networking. Springer-Verlang Berlin, Heidelberg, 2009, s. 13–27.

[3] M. Soysal, E.G. Schmidt: An accurate evaluation of machine learning algorithms for flow-based P2P traffic detection. In *22nd International Symposium on Computer and Information Sciences*, 2007.

[4] Paul Baran: *Paul Baran invents packet switching*. `http://www.livinginternet.com/i/ii_rand.htm`.

[5] Q. Zhang: Understanding the Power of Pull-based P2P Streaming Protocol: We can do even better. In *ACM SIGCOMM*, 2007.

[6] S. Sen, O. Spatscheck, D. Wang: Accurate, scalable in-network identification of p2p traffic using application signatures. In *the 13th international conference on World Wide Web*, ACM, 2004, s. 512–521, iSBN 1-58113-844-X.

[7] Wikipedia: Peer-to-peer — Wikipedia, The Free Encyclopedia. `http://en.wikipedia.org/wiki/Peer-to-peer`, [Online; accessed June-2009].

[8] WWW stránky: Brian's BitTorrent FAQ & Guide. `http://www.btfaq.com/serve/cache/25.html`.

[9] WWW stránky: eMule. `http://www.iana.org/assignments/port-numbers`.

[10] WWW stránky: FastTrack repozitár. `http://cvs.berlios.de/cgi-bin/viewcvs.cgi/gift-fasttrack/giFT-FastTrack`.

[11] WWW stránky: IANA - čísla portov. `http://www.iana.org/assignments/port-numbers`.

[12] WWW stránky: Indiana University. `http://kb.iu.edu`.

[13] WWW stránky: International Telecommunication Union. `http://www.itu.int`.

[14] WWW stránky: Iptables tutorial 1.2.2. `http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html`.

[15] WWW stránky: JFreeChart. `http://www.frozentux.net/iptables-tutorial/iptables-tutorial.html`.

[16] WWW stránky: L7-filter. http://l7-filter.sourceforge.net.

[17] WWW stránky: Linux home networking.
http://www.linuxhomenetworking.com/wiki/index.php/Quick_HOWTO.

[18] WWW stránky: NetFilter. http://www.netfilter.org.

[19] WWW stránky: Switching technology.
http://www.cs.virginia.edu/~mngroup/projects/mpls/documents/thesis/node8.html.

[20] WWW stránky: Traffic Control HOWTO.
http://www.linux.org/docs/ldp/howto/Traffic-Control-HOWTO/intro.html.

[21] Y. Kulbak, D. Bickson: *The eMule Protocol Specification*. DANSS (Distributed Algorithms, Networking and Secure Systems) Lab School of Computer Science and Engineering, The Hebrew University of Jerusalem, Jerusalem, Israel, 2007.

Dodatek A

Instalačný manuál

Testované len na operačnom systéme Ubuntu 9.10 32-bit.
Inštalácia balíčkov potrebných pred inštalovaním nástroja *l7-filter*:

- `sudo apt-get update`
- `sudo apt-get install libnetfilter-conntrack-dev libnetfilter-queue-dev`
- `sudo apt-get install g++`
- `sudo apt-get install conntrackd`
- `sudo apt-get install bind9`

Následne je potrebné stiahnuť a skompilovať nástroj *l7-filter* podľa návodu na [16].

Do programu *iptables* je potrebné vložiť záznamy na povolenie prekladu IP adries. Po spustení prekladania IP adries je potrebné presmerovávať všetky pakety, ktoré sú určené na smerovanie, do nástroja *l7-filter* pomocou následujúceho príkazu:

```
sudo iptables -A FORWARD -j QUEUE --queue 0
```

alebo

```
sudo iptables -A FORWARD -t mangle -j QUEUE --queue 0
```

Ďalej je potrebné pomocou *ant* skriptu preložiť a spustiť aplikáciu:

```
sudo ant compile
```

```
sudo ant run
```

Dodatek B

Konfiguračný návod aplikácie

Aby sme mohli úspešne nájsť peer-to-peer siete a vidieť štatistiky o týchto sieťach, je potrebné mať nasledujúce nastavenia:

1. priradený súbor so vzorom k príslušnému protokolu
2. nastavené správne prístupové cesty k nástrojom: l7-filter, iptables, traffic control
3. nastavené IP adresy, ktoré chceme sledovať

Voliteľné nastavenia:

1. pripojenie do databáze a zároveň ukladanie dát do databáze
2. pridanie blokovaných portov k jednotlivým protokolom
3. pridanie pravidiel, ktoré obmedzujú šírku prenosového pásma danému protokolu

Štatistické grafy je možné pridávať pomocou tlačítka viditeľného v hlavnom okne aplikácie.