VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE PROVOZU V P2P SÍTÍCH ZALOŽENÁ NA MODELU CHOVÁNÍ

SEMESTRÁLNÍ PROJEKT
SEMESTRAL PROJECT

AUTOR PRÁCE Bc. MICHAL ŠPONDR
AUTHOR

BRNO 2008

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE PROVOZU V P2P SÍTÍCH ZALOŽENÁ NA MODELU CHOVÁNÍ

BEHAVIOUR-BASED DETECTION OF P2P NETWORKS

SEMESTRÁLNÍ PROJEKT
SEMESTRAL PROJECT

AUTOR PRÁCE — Bc. MICHAL ŠPONDR
AUTHOR

VEDOUCÍ PRÁCE — Ing. PETR MATOUŠEK, Ph.D.
SUPERVISOR

BRNO 2008

Abstrakt

Tento semestrální projekt se zabývá různými způsoby detekce P2P provozu, hlouběji se zaměřuje na detekci založenou na modelu chování. Porovnává různé způsoby detekce a vysvětluje důvod použití detekce založené na modelu chování. Součástí práce je i náhled na nejčastěji používané P2P protokoly. Navrhuje implementaci detekce neuronovými sítěmi.

Klíčová slova

P2P síť, detekce P2P provozu, neuronová síť

Abstract

This semestral project deals with several detection of P2P traffic, focusing deeper to a behaviour-based detection. It compares various techniques of detection and explains the reason of using behaviour-based detection. This work also includes an examine of most often used P2P protocols. Finally, this work brings in a detection using neural networks.

Keywords

P2P network, P2P traffic detection, neural network

Citace

Michal Špondr: Detekce provozu v P2P sítích založená na modelu chování, semestrální projekt, Brno, FIT VUT v Brně, 2008

Detekce provozu v P2P sítích založená na modelu chování

Prohlášení

Prohlašuji, že jsem tuto diplomovou práci vypracoval samostatně pod vedením pana Ing. Petra Matouška, Ph.D.

. . . . . .
Michal Špondr
6. ledna 2008

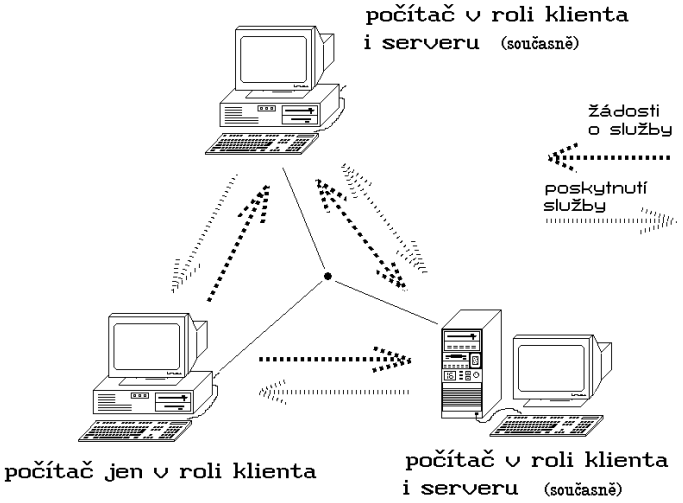
Obsah

Kapitola 1

Úvod

Od počátku existence Internetu si lidé přáli vyměňovat informace a data. K tomu nejdříve sloužila běžná výměna souborů např. přes FTP (a jiné) servery, které však měly nevýhodu – uživatel musel znát jejich umístění (IP adresu). Vznikaly samozřejmě seznamy FTP serverů, uživatelé je však museli neustále ručně aktualizovat, což bylo nepohodlné. Vývoj aplikací pokračoval, s postupným rozšiřováním Internetu mezi uživatele začala být větší potřeba pohodlného vyměňování souborů a tím vznikali i odpovídající P2P sítě, komunikační protokoly a klienti, co se na síť umí připojit.

Co je to vlastně *P2P*? Je to zkratka pro *peer-to-peer* (doslova rovný s rovným) a popisuje architekturu počítačových sítí, ve které spolu komunikují přímo jednotliví klienti (uživatelé). [20] Opačným případem je architektura klient-server, kde uživatelé využívají ke komunikaci s ostatními uživateli nějaký server. Čistá P2P síť pojem server vůbec nezná, komunikují jen klienti; nejčastěji se však používá hybridní přístup, kdy je server sice spojovacím uzlem, ale dále již mezi sebou klienti komunikují sami. Názorně lze vidět peer-to-peer přístup na obrázku 1.1.



Obrázek 1.1: Schéma spojení peer-to-peer, převzato z [14]

P2P sítě jsou jakožto výměnné sítě často spojovány s kriminalitou. Nejčastěji vyměňovaným

materiálem na těchto sítích je totiž hudba, filmy a software, jejichž sdílení bývá často nelegální. Existuje několik společností, které se snaží takové uživatele vypátrat a potrestat. U nás v ČR ještě není autorský zákon [3] pro digitální záznamy na takové úrovni, jako je tomu v západních zemích, proto se občas najdou lidé využívající skulinky v těchto zákonech. Uživatelé dnešních P2P sítí a programů jsou častými hříšníky proti zákonům nebo pravidlům. Presumpce neviny říká jednu věc, zdravý lidský rozum však říká věc jinou.

Další věcí, která může trápit mnoho administrátorů sítí, je fakt, že provoz přes P2P sítě je vysoký, často dlouhotrvající (uživatelé nechávají zapnuté počítače s P2P klienty přes den i noc) a zatěžuje síť na úkor provozu ostatních uživatelů. Celkově pro síť (i když nikoliv pro daného stahujícího či sdílejícího uživatele) je výhodnější dát P2P provozu malou prioritu a zvýhodnit tak datový provoz jiný, u kterého uživatel ,,sedí“ a očekává jeho rychlý a plynulý provoz. Ale zde je třeba dát pozor – P2P sítě sice jsou nejčastěji využívané pro výměnné sítě, není to však jediné využití P2P. Další P2P sítí může být například IP telefonie. Zde by bylo neférové uživatele nějak omezovat, protože ten by mohl zaznamenat kolísání ve kvalitě přenášeného hlasu. Je tedy třeba rozlišit mezi ,,dobrou“ a ,,špatnou“ P2P sítí. Které to jsou, to už je na osobním uvážení.

P2P sítě je tedy třeba nějak detekovat. Detekci provozu je možné provádět více způsoby. Pokud máme detekovánu P2P síť, čili víme, od koho a kam putuje specifický datový tok, je možné s tím něco dělat – snížit prioritu výměnných datových toků, zvýšit prioritu IP telefonie, případně upozornit uživatele na nějaký prohřešek, blokovat provoz zcela atd.

Jak však zjistit, který uživatel sítě se dopouští nekalého či nezákonného P2P provozu? Možností je vícero – buď budeme kontrolovat klasické vlastnosti spojení, jako jsou zdrojový a cílový port, nebo na to půjdeme chytřeji. Tvůrci P2P aplikací jsou si dobře vědomi, že klasický přístup má různé nevýhody. Tím je třeba NAT (Network Address Translation) nebo ,,zlí“ administrátoři snažící se blokovat výměnné sítě. Do protokolů nebo klientů tak zavádí různé dodatečné možnosti, jako šifrování, používání širokého rozsahu portů včetně privilegovaných (není tedy možné rozeznat P2P provoz od webového přenosu jen podle cílového portu). Detekce takového provozu musí tedy být více sofistikovaná, aby dokázala takový provoz bezpečně najít a neohrozit tím provoz jiným uživatelům. Tímto tématem se zabývá např. diplomová práce Oldřicha Plchota [15].

1.1 Historie a typy sítí P2P

Přeskočme prapočátky výměny souborů přes sítě Usenetu a FidoNetu [16], dvou velmi úspěšných a rozsáhlých decentralizovaných sítí uživatelů, které sice také sloužily k výměně souborů, ale nešlo o P2P sítě v dnešním slova smyslu. Výměnné P2P sítě se dají rozdělit na čtyři generace podle typů používaných klientů [17].

První generaci tvoří sítě **Napster** a **Direct Connect**. Ty rozšířily podvědomí o P2P sítích a výměně souborů. Princip Napsteru byl jednoduchý. Stejnojmenná společnost Napster vlastnila centrální server, který indexoval všechny soubory, které uživatelé sdíleli. Vyhledávání souboru pak spočívalo v zaslání dotazu na serveru Napsteru, který poslal zpátky odpověd se seznamem uživatelů. To se ukázalo jako výhodné pro firmu a zároveň výhodné pro právníky, kterým stačilo pouze odpojit server Napsteru od Internetu. DC (Direct Connect) má stejnou střední míru decentralizace jako Napster, také využívá serveru k propojení

uživatelů. Jde o rozšíření komunikačního protokolu IRC o funkce týkající se výměny souborů.

Druhou generaci P2P sítí spustila síť **Gnutella**. Tvůrci byli poučení nevýhodou Napsteru a vytvořili *decentralizovanou* síť. Klienti se po spuštění připojili k určitému množství dalších klientů, kteří byli připojení k jiným klientům atd. Když klient hledal soubor, zeptal se počítačů, ke kterým byl připojen, na daný soubor. Ti, pokud ho měli, ho poslali zpátky, pokud ne, poslali dotaz dalším klientům, na které byli připojeni. Nevýhodou této sítě byla její pomalost a také ostrůvky sítí, které mezi sebou nebyly propojeny.

Třetí generace stojí na P2P síti **Fasttrack**, kterou využívají klienti jako Kazaa, Grokster, Morpheus. Tato síť byla postavena na stejném principu jako Gnutella, avšak využívala navíc vylepšení jako superuzly a pokračování ve stahování souborů. Zajímavostí je, že na vzniku protokolu Fasttrack se podílel stejný tým lidí, který vymyslel Skype.

Čtvrtou generací by mohl být systém **Bittorrent**. Ten je v současnosti nejvyužívanější P2P sítí. Základní myšlenkou bylo ulehčení práce hlavního serveru, ze kterého má být soubor stahován a přenesení této zátěže na uživatele. Pro zpřístupnění informací o uloženém souboru slouží tzv. *trackery* (sledovače), což jsou počítače, které uchovávají IP adresy uživatelů. Uživatel se k tomuto trackeru připojí, získá informace o dalších uzlech (a sám je jeho podíl na výměně zaznamenán v trackeru) a od těch poté stahuje po malých dílech soubor, stejně tak je i od něj soubor stahován jinými klienty. Výhodou je tedy menší zatížení serveru, decentralizace. K nevýhodám patří fakt, že touto technologií lze stahovat pouze aktuální žádané soubory – v případě, že soubor není žádaný, sdílí jej stále méně lidí a síť tak zanikne.

Podívejme se na P2P sítě, které k výměně souborů neslouží. Mezi nejtypičtější další využití je bezesporu IP telefonie. K nejpoužívanějším zástupcům IP telefonie je síť **Skype**, i když nejde o VoIP v původním slova smyslu. Tato síť má pozitivní i negativní vlastnosti. K pozitivům patří, že se dokáže protlačit přes firewally i NAT. Dělá to však takovým způsobem, který se dá brát za její negativum – jak již bylo řečeno výše, podíleli se na její tvorbě autoři Fasttracku, využívá podobnou technologii, tedy superuzly, přes které teče datový tok i bez vědomí uživatele nebo administrátora sítě. A protože může využívat ke svému provozu i nejčastěji používaný port 80, tzn. www, lze ji těžko blokovat, musí se dělat rozbor paketů až na aplikační úrovni ISO/OSI. Někde je třeba blokovat a tudíž předtím i rozpoznat tento datový tok (například pokud někdo platí za přijatá/odeslaná data, nemusí být pro něj tento provoz výhodný).

Kapitola 2

Architektura sítí P2P

Podívejme se detailněji na princip a komunikační protokoly některých typických P2P sítí. Jejich pochopení napomůže k vysvětlení principů detekce, kterým se budeme věnovat v kapitole 3 na straně 12).

P2P sítě se dají dělit podle několika kritérií [21]. Nejčastěji se dělí podle způsobu použití:

- Sdílení souborů (zmíněno v úvodu – DC, Bittorrent, Fasttracker, ...)
- IP telefonie (Skype, SIP)

Další klasifikací je rozdělení podle **stupně centralizace**. V „*čistých*“ P2P sítích si jsou všechny uzly rovny (neexistuje dělení na klienta a server), neexistuje centrální server spravující síť ani žádný jiný centrální server. Nejtypičtějším zástupcem čisté P2P sítě je Gnutella a Freenet (případně Usenet a FidoNet, to však nejsou přímo sítě pro výměnu souborů). Existuje však velký počet hybridních systémů, jelikož čistý P2P přístup se neukázal jako zcela výhodný (pomalost šíření informací, ostrůvky osamostatněných sítí atd.) K typickým *centralizovaným* P2P sítím patří například zaniklý Napster.

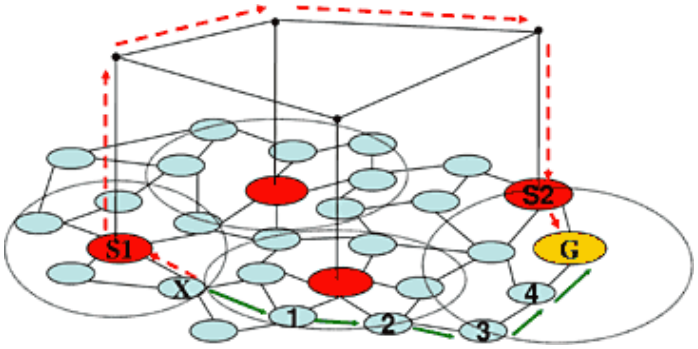
Sítě je dále možné dělit na **strukturované** a **nestrukturované**.

2.0.1 Strukturované a nestrukturované sítě

Překryvná síť (angl. *overlay network*)[1] sestává z uzlů. Tyto uzly jsou propojeny s jinými uzly (ne nutně každý s každým) – podle toho, jak jsou tyto uzly propojeny, dělíme síť na strukturovanou a nestrukturovanou. Schéma překryvné sítě a znázorněná cesta v ní je na obrázku 2.1.

Nestrukturovaná síť vzniká v případě, že je překryvná síť vytvářena libovolným způsobem. Typicky když se nějaký klient připojí, získá od sousedů jejich data a ty pak modifikuje dle svého uvážení. V případě, že chce klient získat nějaký soubor, je dotaz šířen sítí k co nejvíce uživatelům. Hlavní nevýhodou této sítě je, že ne vždy je dotaz vyřešen. Oblíbený obsah bude častěji dostupnější, zatímco vzácný obsah nemusí být nalezen, i když třeba někde v síti je, jen k němu neexistuje cesta. Vysoké množství řidícího datového toku

[1] Jde o síť vystavenou nad jinou sítí, uzly v překryvné síti jsou spojeny virtuálními či logickými spoji, které odpovídají cestě, která zase může být tvořena více fyzickými cestami. V případě P2P sítí tedy mnoho sítí tvoří překryvnou síť nad Internetem.



Obrázek 2.1: Schéma překryvné sítě, převzato z [1]

snižuje šířku pásma pro samotnou výměnu souborů. Nejčastějšími strukturovanými P2P sítěmi jsou Gnutella a Fasttrack.

Strukturovaná síť používá komunikační protokol využívající směrování a umožňující tak najít jakémukoliv uzlu cestu k jinému uzlu majícímu požadovaný soubor, i když je velmi vzácný. Tato záruka vyžaduje strukturovanější přístup překryvné sítě. Nejčastějším typem strukturované sítě je *distribuovaná hashovací tabulka* – distributed hash table (DHT). Zde se používá konzistentní hashování[2] pro přiřazení souboru s daným uživatelem. Mezi strukturované sítě využívající DHT patří Chord, Pastry, Tapestry, CAN či Tulip.

Podívejme se na některé nejznámější P2P sítě podrobně.

2.1 Komunikační protokoly sítí P2P

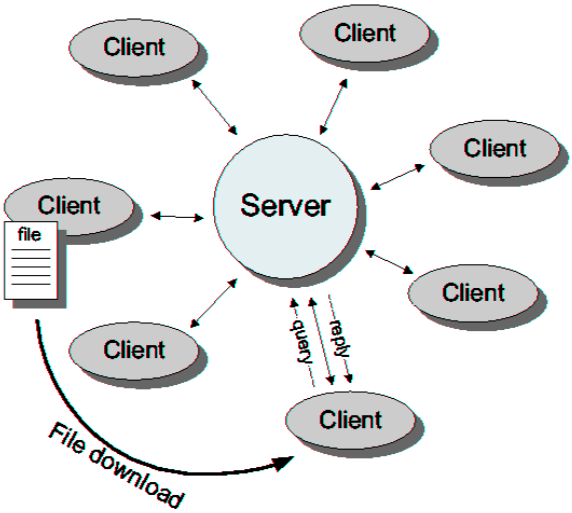
2.1.1 Direct Connect

Direct Connect (DC) protokol vznikl v roce 1999 [19]. Jde o centrální P2P systém vycházející z IRC (Internet Relay Chat) obohacený o funkce výměny souborů. Stejně jako IRC obsahuje *hub*y, což je anglický termín pro centrální servery, kam se uživatelé přihlašují. Huby nabízejí informace o klientech, čímž umožňují vyhledávání souborů a chatování. Přenos souborů jde čistě přes klienty. Schéma spojení účastníků sítě Direct Connect je podobné, jako u Napsteru (tam však existoval pouze jeden hlavní server) a vypadá jako na obrázku 2.2.

DC protokol je textový, příkazy jsou zasílány bez šifrování. Od připojení hubu je vyžadováno, aby měl dokázal rychle odesílat data, jelikož se na něj bude připojovat mnoho uživatelů vyžadujících nějakou informaci. *Neexistuje* žádná oficiální specifikace protokolu. Současný protokol vznikl reverzním inženýrstvím klienta Neo-modus. Proto se mnoho klientů může od sebe odlišovat.

Po připojení k socketu serveru klientem začíná komunikaci na aplikační úrovni jako první server. Protokol také nespecifikuje kódování, které má klient či server využívat. Standardním portem pro komunikaci hubů je port 411, pro komunikaci mezi klienty 412.

[2] To je takové hashování, kde přidání či odebrání nějaké hodnoty výrazně nezmění mapování klíčů na hodnoty; průměrně je při změně tabulky potřeba přemapovat $\frac{k}{n}$ klíčů (k – počet klíčů, n – počet hodnot) a ne celou tabulku, jak je tomu v případě normální hashovací tabulky.



Obrázek 2.2: Schéma sítě Direct Connect, převzato z [5]

V případě, že jsou tyto porty obsazeny, používá se port o 1 vyšší. Uživatelé jsou na hubu identifikováni svými přezdívkami, neexistuje žádné globální rozlišování přezdívek. Příchozí požadavek na komunikaci klient-klient nemůže být spojen s aktuálním připojením, podobně výsledek vyhledávání nemůže být spojen s nějakým jiným konkrétním výsledkem vyhledávání. Protokol podporuje vyhoštění uživatele z hubu (tzv. kick) či přesměrování uživatele na jiný hub (neexistuje pravidlo, kam to bude). Pokud je uživatel vyhoštěn a odpojen, hub mu nemusí vypsat důvod, v případě přesměrování musí. Co se přesměrování týče, neexistuje ekvivalent HTML referreru, čili hub neví, zda na něj byl nějaký uživatel přesměrován a odkud. Hub může klientovi zaslat uživatelský příkaz, jde však jen o příkaz na uživatelské úrovni (nelze tedy například spustit klientovi prohlížeč).

P2P část protokolu je založena na *slotech*. Slot značí, kolik uživatelů může od daného klienta v jeden okamžik stahovat. Počet slotů je volen klientem. Na začátku klient-klient komunikace si jednotlivé strany vymění náhodné číslo, ten kdo má větší, může stahovat jako první. Stahování probíhá na protokolu TCP, vyhledávání používá UDP. Připojení k hubu je přes TCP. Klienti mohou být ve dvou módech: v aktivním nebo pasivním. Aktivní uživatel může stahovat od kohokoliv na síti. Pasivní uživatelé mohou stahovat pouze od aktivních. Pasivní uživatelé dostanou výsledky od hubu, aktivní přímo od ostatních. Aktivní uživatel tedy naslouchá na TCP a UDP portu.
Oddělovače protokolu jsou `\$`, `|` a ␣ (mezera). Není možné poslat escape sekvenci, takže tyto znaky není možné poslat jako zprávu.
Do protokolu byla dále přidána podpora broadcastového šíření Tiger-Tree hashů (TTH)[3] sdílených souborů. Tím lze kontrolovat, zda byl soubor stažen správně, a hledat stejné soubory nezávisle na jejich názvu.

Další informace lze získat na z dokumentace [2] projektu Open Direct Connect, kde je popsána i dokumentace protokolu. Jak již však bylo napsáno, nejde o oficiální dokumentaci.

[3] TTH – http://en.wikipedia.org/wiki/Hash_tree#Tiger_tree_hash

2.1.2 Bittorrent

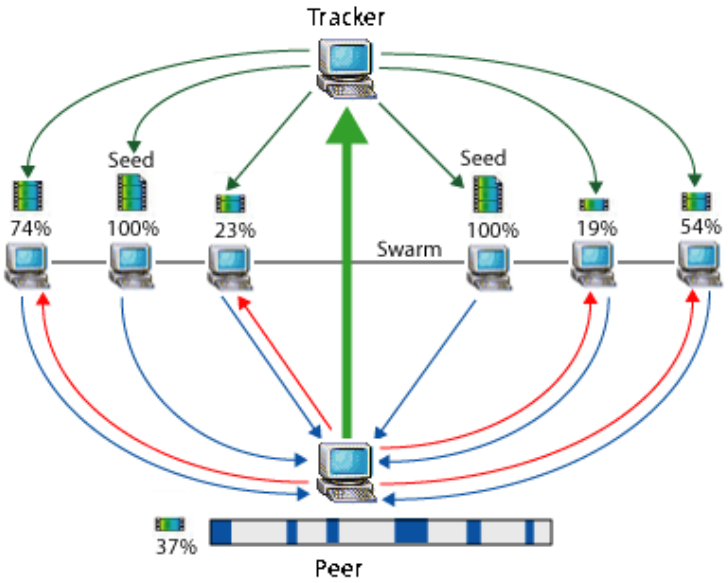
Bittorrent je protokol navržený pro distribuci velkého objemu dat a zároveň ušetření prostředků původního zdroje. Princip sdílení dat je založen na rozložení zátěže stahování na všechny účastníky provozu. Každý účastník poskytuje ostatním kousky dat (ta tedy nejsou stahována jen z původního zdroje), data jsou redundantní[4] a zabraňuje se tím nemožnosti stahovat v případě výpadku jednoho stroje – systém je stabilnější. Autorem protokolu je Bram Cohen, vytvořil ho v roce 2001 a v současnosti je použití Bittorrentu nejspíše největší využití sítě Internet (CableLabs – 18 % provozu internetu[5]).

Ke sdílení dat je třeba nejprve vytvořit tzv. *torrent* (v překladu „proud“). Jde o malý soubor obsahující metadata (popis níže) o sdíleném souboru a o tzv. *tracker*u, tedy počítači řídícímu distribuci dat. Uživatel (v terminologii Bittorrentu se mu říká *peer*) musí tedy nejdříve sehnat daný torrent a připojit se k trackeru, který mu řekne, od kterých dalších připojených uživatelů může stahovat kousky dat. Oproti běžně používanému HTTP požadavku stahuje Bittorrent jinak: zatímco HTTP požadavek GET stahuje data přes jeden TCP socket, Bittorrent vytváří mnoho P2P požadavků přes vícero socketů. HTTP požadavek stahuje soubor sekvenčně, Bittorrent stahuje náhodně nebo stylem „prvně nejvzácnější“ (anglický termín rarest-first je snad výstižnější). Tím se snižují výsledné náklady, zvyšuje redundance a odolnost vůči chybám, na druhou stranu může trvat delší dobu, než klient vytvoří dostatek spojení. Také s touto metodou nelze média streamovat, i když B. Cohen naznačil[6], že streamovatelné torrenty jsou jen otázkou času.

Uživatel distribuuje data po kouscích o velikosti 64 kB – 1 MB. Kus o velikosti vyšší než 512 kB zmenší pro velká data velikost torrent souboru, avšak snižuje se účinnost protokolu [18]. Peer vytvoří pomocí hashovacího algoritmu ke každému kousku kontrolní součet a vloží jej do torrent souboru. Ten si budou znovu vypočítávat ostatní peer uživatelé, jakmile kousek souboru získají. Peer uživatelé nabízející celý kompletní soubor se nazývají *seeder*. Přesný obsah torrent souboru je závislý na verzi Bittorrent protokolu. Konvencí je používat příponu .torrent. Dále torrent soubory obsahují sekci s názvem announce, která oznamuje URL trackeru, sekci info obsahující (doporučené) názvy souborů, jejich velikosti, použitou délku dílků a SHA-1 kód pro každý kousek. Torrent soubory jsou umístěny například někde na webové stránce a jsou zaregistrovány v trackeru. Tracker uchovává seznamy uživatelů právě stahujících (či sdílejících) torrent. Existuje ještě beztrackerový systém, kde se každý peer chová jako tracker; zde se používá ve většině klientů metod distribuovaných hashovacích tabulek.

Stahující uživatel si najde a stáhne torrent, otevře jej ve svém Bittorrent klientovi, ten se připojí k trackeru, odkud získá seznam ostatních klientů. Klient se připojuje k ostatním a žádá po nich kousky souboru. Skupina peerů se nazývá *swarm* (český překlad „hejno“ se ani v našich krajích nepoužívá). Pokud swarm obsahuje pouze počátečního klienta nabízející celý soubor, klient se k němu připojí a začne stahovat dílky, jinak od sebe peer uživatelé stahují i od sebe navzájem. Klienti se snaží optimalizovat své poměry downloadu a uploadu; například že stahují náhodná data, aby se zvýšila šance na další výměnu dat, což je možné, jen pokud mají dva klienti jiné kousky souboru. Efektivita výměny dat také závisí na po-

[4] Redundance je stav či vlastnost, kdy je použito větší množství prvků než je obvyklé nebo nutné.
[5] http://www.multichannel.com/article/CA6332098.html
[6] http://torrentfreak.com/interview-with-bram-cohen-the-inventor-of-bittorrent/



Obrázek 2.3: Schéma sítě Bittorrent, převzato z [8]

litice klientů rozhodujících se, komu posílat data. Klient může preferovat zasílání dat jen těm, kteří zasílají data jemu[7]. Přísná politika však může vést k neoptimálním výsledkům, např. nový připojený klient nemůže nic stáhnout, protože nemá co nabídnout nebo když ani jeden ze dvou klientů, byť s dobrým připojením, nezačne druhému nic nabízet. Oficiální Bittorrent klient používá mechanismus ,,optimitického uvolňování“, kdy si ponechá část dostupné šířky pásma pro zasílání dílků náhodným uživatelům v naději, že nalezne ještě lepší partnery, a tím umožňuje nově příchozím zapojit se do swarmu.

Bittorrent svému uživateli nezaručuje anonymitu. Lze zjistit IP adresy všech současných (a někdy i předchozích) účastníků swarmu v trackeru. Nevýhodou Bittorrentu je i to, že mnoho účastníků se po stažení dat z trackeru odpojí. Tím swarm postupně odumírá. Proto je Bittorrent vhodný na přenos aktuálních a žádaných souborů, starší nemívají dostatek klientů tyto soubory nabízejících. Bittorrent je také málo výhodný v prostředí vytáčených linek a telefonních připojení, jelikož ty mívají nižší poměr uploadu vůči downloadu, klient tak není schopen nabídnout ostatním ke stažení to, co je schopen stáhnout on sám (často jsou takoví uživatelé nazýváni anglicky *leechers*).

Jak vypadá schéma sítě Bittorrent je možno vidět (včetně znázornění používaných termínů) na obrázku 2.3. Konkrétnější informace o protokolu, které však nejsou pro náš případ nezbytně nutné, lze naléz na webových stránkách [18], [4].

2.1.3 Skype

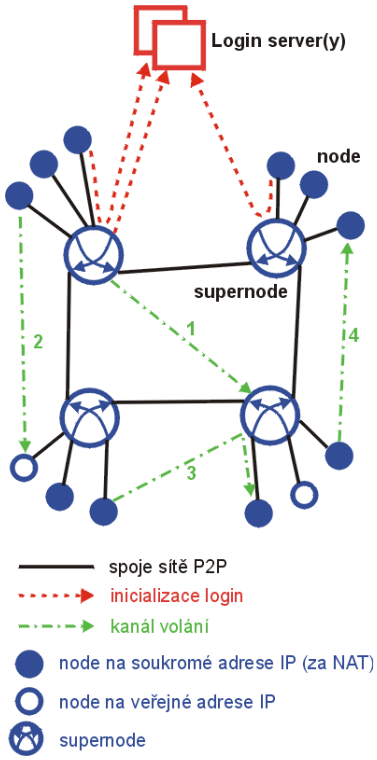
Skype je proprietární protokol internetové telefonie (VoIP). Protokol nebyl zveřejněn, není tedy znám jeho přesný popis, reverzním inženýrstvím však byly zjištěny alespoň nějaké

[7] strategie ,,oko za oko“ – http://en.wikipedia.org/wiki/Tit_for_tat

informace [22].

Síť Skype se skládá ze tří hlavních entit: superuzlů (angl. supernode), běžných uzlů (angl. node) a serveru. Jde o překryvnou síť, každý klient vytváří a udržuje seznam dostupných uzlů známých jako *host cache*. Tato cache obsahuje IP adresy a porty superuzlů. Komunikace je šifrovaná pomocí RC4[8], protože jde však o jednoduchou šifru, slouží spíše ke skrytí provozu.

Skype klient se stává automaticky superuzlem v případě, že má k dispozici dobrou šířku pásma, není za firewallem a má dostatečně výkonný procesor. Superuzly se shlukují do *slot*ů (9–10 superuzlů). Sloty se shlukují do *blok*ů (8 slotů). Skype umožňuje díky superuzlům volání i uživatelům za symetrickým NATem[9] a firewallem. To však zatěžuje superuzly, protože přes ně jde většina provozu.
Schéma Skype sítě je zobrazeno na obrázku 2.4.



Obrázek 2.4: Schéma sítě Skype, převzato z [13]

Přihlašování

Klient autentizuje uživatele na přihlašovacím serveru, rozšíří informaci o své přítomnosti dalším připojeným uživatelům, zjistí typ NAT či firewallu, za kterým se nachází a začne

[8] RC4 – http://en.wikipedia.org/wiki/RC4

[9] Symetrický NAT je takový, který každý požadavek ze stejné interní IP adresy a portu na určitou IP adresu a port namapuje jedinečnou externí zdrojovou IP adresu a port. Pokud stejný vnitřní uživatel pošle byť ze stejné IP adresy a portu požadavek, avšak jinému cíli, pak je použito jiné mapování. Paket smí zaslat zpět jen ti vnější uživatelé, kteří obdrželi paket od vnitřního uživatele.

objevovat uzly s veřejnou IP adresou. Pro připojení ke Skype síti musí host cache obsahovat platnou položku. Musí být ustanoveno spojení se superuzlem, jinak se přihlášení nepovede.

Zde je pseudokód přihlášení se na Skype server:

```
1. start
2. zašli UDP paket(y) na host cache
3. if nedojde do 5 sekund odpověď then
4.  pokus se o TCP spojení s host cache
5.  if nejsi připojen then
6.   zkus TCP připojení s host cache na portu 80 (HTTP)
7.   if nejsi připojen then
8.    zkus TCP připojení s host cache na portu 443 (HTTPS)
9.    if nejsi připojen then
10.    pokusy++
11.    if pokusy==5 then
12.     fail
13.    else
14.     čekej 6 sekund
15.     goto 2
16.Success
```

Po připojení se musí klient autentizovat uživatelským jménem a heslem ke Skype serveru. Těch je celá řada, využívají různé porty a jsou zašifrovaně zapsány přímo do binárního souboru Skype.
Při každém přihlášení vytvoří Skype klíč relace o délce 192 náhodných bitů. Klíč je zašifrován 1536–bitovým šifrovacím klíčem (který je také zakódován přímo v aplikaci). Také se vytvoří 1024-bitový privátní a veřejný pár RSA klíčů. Dále je vytvořen MD5 hash ze spojení uživatelského jména, řetězce `Skyper\n` a hesla jako sdílené tajemství s přihlašovacím serverem. Nešifrovaný klíč relace je zašifrován 256-AES klíčem, který je použit pro zašifrování veřejného RSA klíče relace a sdíleného tajemství. Zašifrovaný klíč relace a hodnota AES je poslána přihlašovacímu serveru.

Server získá klíč relace dešifrováním zašifrovaného klíče relace svým privátním RSA klíčem. Klíč relace je pak použit pro dešifrování veřejného RSA klíče relace a sdíleného tajemství. Pokud sdílené tajemství souhlasí, podepíše server veřejný RSA klíč uživatele svým privátním klíčem. Podepsaná data jsou zaslána superuzlům. Při hledání uživatele vrátí superuzel jeho podepsaný veřejný klíč. Skype klient uživatele autentizuje a domluví se na klíči relace pomocí zmíněného RSA klíče.

Při provozu jsou běžné TCP/UDP pakety obsahující datové části šifrovány pomocí RC4 a mohou být také šifrovány aritmetickým kódováním[10]. Skype je však stále ne zcela prozkoumaná oblast, proto je jeho datový tok na síti celkem obtížně rozeznatelný.

[10] Aritmetické kódování je metoda pro bezztrátovou kompresi dat podobnou Huffmanovu kódování; viz http://en.wikipedia.org/wiki/Arithmetic_coding

Kapitola 3

Detekce komunikace sítí P2P

V předchozí kapitole jsme si rozebrali různé typy sítí P2P. Každá z těchto sítí používá určitý protokol, ať už otevřený nebo proprietární. Jak jsme si popsali v kapitole Úvod na straně 2, vyskytne se někdy potřeba síť z nějakého důvodu detekovat. Uživatel, který má na počítači spuštěný jen P2P klienta, žádný jiný síťový program mu tam neběží, má jasno: jediný datový tok směřující z/do jeho počítače, je ten P2P provoz (samozřejmě pokud se nevyskytne nějaká extrémnější situace, jako pokus o scanování portů apod.) Co ale v případě, že na počítači běží více síťových programů? A nebo ještě výstižnější příklad – mějme nějakou bránu (gateway) pro uživatele malé sítě, přes kterou datový tok jen prochází. Jak zjistíme, zda se v onom datovém toku vyskytuje P2P provoz a pokud tam je, od koho a kam směřuje? Odpovědí na tuto otázku budou následující odstavce.

Jak probíhá taková detekce síťového provozu obecně? V první řadě je třeba datový tok nějak zachytit. Přesněji zachytit a nepozměněný přeposlat dále (*sniffing*). V dalším textu budu uvažovat výhradně sniffing na transportní (TCP) či síťové (IP) vrstvě, nižší vrstvy nejsou uvažovány.

3.1 Detekce založená na informacích z hlaviček paketů

Detekce založená na čtení informací z hlaviček paketů patří k těm nejjednodušším, co se návrhu týče. Typická TCP/IP hlavička vypadá jako na tabulce 3.1. Detekce provozu

0		8	16	31	
Verze IP	Délka záhlaví	Typ služby	Celková délka IP datagramu		IP
Identifikace IP datagramu			Příznaky	Posunutí fragmentu od počátku	
Doba života datagramu		Protokol vyšší vrstvy	Kontrolní součet z IP záhlaví (checksum)		
IP adresa odesílatele (*source IP adress*)					
IP adresa příjemce (*destination IP adress*)					
Volitelné položky IP hlavičky					
Zdrojový port (*source port*)			Cílový port (*destination port*)		TCP
Pořadové číslo odesílaného bytu					
Pořadové číslo přijatého bytu					
Délka hlavičky	Rezerva	U A P R S F	Délka okna		
Kontrolní součet			Ukazatel naléhavých dat		
Volitelné položky TCP hlavičky					
DATA					

Tabulka 3.1: IP a TCP hlavička [6]

založená na informacích z hlaviček paketů by spočívala ve znalosti IP adresy odesílatele a příjemce a zdrojového a cílového portu. Kterou IP adresu a port vybíráme, závisí na tom, co vlastně hledáme.
Pokud hledáme někoho připojujícího se k dané službě, hledáme IP adresu odesílatele (IP adresa příjemce nám k ničemu není, pokud nemáme seznam vytipovaných IP adres, na kterých tušíme, že nějaká služba běží) a cílový port. Podle cílového portu poznáme, kterou službu, v našem případě P2P službu, si přeje odesílatel (s danou IP adresou) využít. Tyto dvě informace tedy určují *uživatele* P2P služby.
Pokud hledáme nějakou P2P službu, ke které se připojují uživatelé, a která běží na námi hledané síti, nezáleží nám moc na tom, jací konkrétně uživatelé to jsou, zajímá nás daný stroj se službou. Proto se zaměříme na hledání IP adresy odesílatele a zdrojový port odesílatele. Tyto dvě informace určují *zdroj* P2P služby.
Zbývá tedy zjistit, jaké porty využívají ke své činnosti vybrané P2P služby.

3.1.1 Detekce podle portu

Jde o nejzákladnější a přímou metodu pro zjištění uživatelů na síti využívajících P2P služeb. Je založena na faktu, že mnoho aplikací používá pro svůj běh standardní porty. Příkladem je třeba (převzato z [10]):
Limewire 6346/6347 TCP/UDP
Morpheus 6346/6347 TCP/UDP
BearShare default 6346 TCP/UDP
Edonkey 4662/TCP
EMule 4662/TCP 4672/UDP
Bittorrent 6881–6889 TCP/UDP
WinMx 6699/TCP 6257/UDP

Administrátorovi při detekci P2P provozu podle portů stačí vyhledat spojení používající tyto porty. Jde o jednoduchý způsob, jeho nevýhody jsou však zjevné: většina P2P programů umožňuje změnit uživateli standardní port, navíc některé používají porty náhodně. Objevuje se také trend používat tunelovaná spojení, tedy taková, která se skrývají za portem jiných, většinou privilegovaných, služeb.

3.1.2 Detekce analýzou protokolu

Další možností je analýza protokolu na aplikační vrstvě. Administrátor (respektive aplikace, kterou při tom použije) se musí podívat do datové části paketu a porovnat si s nějakými vzorky již známých P2P signatur. K tomu slouží aplikace jako L7-filter, PDML, netscreen-IDP, NetScout aj. Ty testují, zda daná získaná data souhlasí s regulárním výrazem signatury. [9]

Zde prohrávají boj P2P sítě spoléhající na konkrétní port, jak bylo popsáno v předchozí části. Výsledek je přesnější, avšak má svoje neduhy. Protože se P2P aplikace vyvíjí, mění se i signatury. Je třeba stále obměňovat a doplňovat signatury. Navíc se provoz P2P sítí se stále více přesouvá do šifrované sféry a pokud (vezměme v úvahu šifrování přes SSL) by byla provedena analýza na aplikační vrstvě, došlo by k porušení integrity dat. Další nevýhodou je, že analýza provozu na velké síti s velkými objemy dat by mohla vést k její nestabilitě. Pokud by totiž aplikace selhala, co by se stalo se síťovým provozem? A konečně analýza na aplikační vrstvě je vysoce náročná na dostupné hardwarové prostředky. Data je třeba

zkoumat za běhu, aniž by uživatel pocítil nějaké výrazné zpomalení, což v případě 1Gbit či 10Gbit sítí není levné.

3.2 Detekce založená na chování datového toku

Tento způsob detekce využívá nějaké *statistické* nebo *heuristické analýzy* datového toku. Nezajímá se tedy o konkrétní obsah paketů, ale o celkové chování toku, jeho velikosti, velikosti paketů a podobně.

Informace o síťovém provozu lze zjistit bez výrazného zatížení z různých síťových zařízení na síti. Na menších sítích stačí pozorovat chování na bráně (gateway), na větších sítích mohou administrátoři povolit na svých routerech a přepínačích funkci *Netflow* pro export údajů o síťovém provozu.

Ke známému způsobu detekce P2P provozu patří dle [12] zkoumání následujícího chování:

- Využití TCP i UDP ve vymezeném časovém rámci mezi stejnou zdrojovou a cílovou IP adresou
- Připojení k většímu počtu odlišných portů než je počet různých IP adres

V článku [7] se objevila zmínka o identifikaci datového toku (ne přímo P2P provozu) založenou na třech různých úrovních. *Sociální* úroveň je tvořena interakcí mezi koncovými stanicemi, *síťová* úroveň analyzuje roli jednotlivých stanic (producent, konzument či obojí) a *aplikační* úroveň analyzuje interakci mezi koncovými stanicemi na daných portech, aby tak rozpoznala používané aplikace.

Výhody a nevýhody

Metoda detekce P2P sítí na základě chování může být využita s výhodou tam, kde je z důvodů legálnosti, soukromí nebo technické nemožnosti (šifrování) nasazení detekce analýzou protokolu nemožné. Zpracování dat může být odloženo, data lze ukládat do databází a klasifikační algoritmus lze použít později. Zpracování analýzou protokolu by nebylo snadné, jelikož bychom museli uchovávat veškerá data, ne pouze informaci o provozu. Dále lze detekci na základě chování využít na vícenásobná monitorovací místa, analýza protokolu neumožňuje takovouto agregaci dat. Dále je metodu detekce podle chování možno využít i pro neznámé P2P sítě, zatímco detekce analýzou protokolu by musela provádět reverzní inženýrství tohoto protokolu (případně pátrat v dokumentaci, obojí je zdlouhavé a netriviální).

Nevýhodou detekce na základě chování je, že neřekne konkrétní informace (např. použitý P2P klient), někdy může být informace přibližná až mylná (např. v případě neuronových sítí, které nejsou zcela dobře naučeny). Navíc zázemí pro analýzu protokolu může být použito pro více účelů, jako je detekce virů, červů apod., analýza na základě chování se opírá o specifické vlastnosti, které se jinde využít tak snadno nedají.

3.2.1 Detekce podle UDP paketů

Ukázalo se [10], že P2P aplikace mají zajímavé chování UDP spojení, které lze využít pro detekci uživatelů používajících P2P aplikace s decentralizovanými sítěmi. To jsou v současnosti

nejrozšířenější sítě, protože na první úspěšnější P2P síti Napster se ukázalo, že centralizovaný přístup není výhodný. Jak pracuje decentralizovaná síť bylo popsáno na str. 4. Každý uživatel (peer) se musí o ostatních peerech dozvědět, potřebuje komunikovat se superuzly, vyhledávat soubory, zjišťovat stav ostatních atd. Je tedy třeba nějakých *řídicích paketů*, které se o tyto činnosti postarají. K tomu poslouží nejlépe právě UDP. Proč? UDP je jednoduchý, efektivní a nenáročný. Nepotřebuje potvrzení o doručení, vytvoření spojení nebo udržení připojení.

Zjistilo se [10], že víceméně všechny decentralizované P2P aplikace vykazují podobné chování na síti. Po spuštění vytvoří jeden či více UDP socketů pro naslouchání a komunikaci s velkým počtem vnějších adres. Podívejme se na pročištěný seznam UDP spojení P2P aplikace Edonkey[1]:

```
11:24:19.650034 IP x.10810 > y.34.233.22.8613: UDP, length: 25
11:24:19.666047 IP x.2587 > y.138.230.251.4246: UDP, length: 6
11:24:19.666091 IP x.10810 > y.127.115.17.4197: UDP, length: 25
11:24:19.681433 IP x.10810 > y.76.27.4.4175: UDP, length: 25
11:24:19.681473 IP x.2587 > y.28.31.240.4865: UDP, length: 6
11:24:19.696907 IP x.2587 > y.162.178.102.4265: UDP, length: 6
......
11:24:20.946921 IP x.2587 > y.250.47.34.4665: UDP, length: 6
11:24:20.962509 IP x.2587 > y.152.93.254.4665: UDP, length: 6
11:24:20.978275 IP x.2587 > y.28.31.241.5065: UDP, length: 6
11:24:20.993871 IP x.2587 > y.135.32.97.580: UDP, length: 6
11:24:21.009621 IP x.2587 > y.149.102.1.4246: UDP, length: 6
11:24:29.681224 IP x.10810 > y.32.97.189.5312: UDP, length: 4
11:24:29.696903 IP x.10810 > y.10.34.181.7638: UDP, length: 4
11:24:29.716503 IP x.10810 > y.26.234.251.12632: UDP, length: 4
......
11:26:20.291874 IP x.10810 > y.19.149.0.21438: UDP, length: 19
```

Na výstupu si lze všimnout, že datový tok pochází ze dvou zdrojových portů, UDP 2587 a UDP 10810 (tyto porty Edonkey náhodně vytvořil). Cílové adresy se liší.

Popis tohoto vzorku chování se dá popsat následovně: Pokud za čas $t(x)$ vznikne z jedné IP adresy a daného UDP portu spojení na y IP adres s pevnými nebo náhodnými UDP porty a $x = 5$, $y = 3$, pak lze říct, že se v datovém toku vyskytuje P2P provoz. Administrátor samozřejmě může tyto hodnoty upravit dle svých potřeb. Existují samozřejmě i zde výjimky. Tou může být například herní server, DNS server nebo server streamující média. U těchto případů však často lze rozeznat rozdíl, jelikož server běžně nevytváří v takové míře spojení na jiné porty, než je jeho funkční port, což však dělá model decentralizované P2P sítě. Výhodou tohoto řešení tedy je, že se nemusí spoléhat na data z aplikační vrstvy ani na signatury (nově vzniklé P2P sítě tak mohou být rychle odhaleny). Nevýhodou je, že tímto způsobem lze zjistit pouze decentralizované sítě (i když to je dnes většina P2P sítí). Toto řešení navíc neplatí na síť Bittorrent z důvodu použití trackerů. Dále se nedá použít tam, kde je použit protokol TCP.

[1] http://en.wikipedia.org/wiki/EDonkey_network

Identifikace P2P aplikací

Ještě podrobnější analýzou UDP provozu P2P aplikací lze zjistit další zajímavé vzorky chování. Řídicí pakety mají často stejnou velikost. Na základě tohoto a specifického chování konkrétních P2P sítí lze zjistit typ aplikace, která P2P provoz vytváří [10].

Edonkey2000 rozesílá po startu mnoho zpráv ‚*požadavek na stav serveru*' o velikosti 6 B. Neustále se objevují prohledávací pakety o velikosti 25 B.

BearShare po startu vyšle na mnoho různých míst pakety o délce 28 B. Pokaždé, když je spuštěn přenos souboru, je odesílateli souboru zasíláno velké množství UDP paketů o velikosti 25 B.

Limewire po startu zasílá mnoho paketů o velikosti 35 B a 25 B. Při spuštění stahování souboru je ven zasláno mnoho UDP paketů o velikosti 23 B.

Skype začíná rozesíláním 18 B velkých UDP paketů.

Kazaa po startu rozešle paket o velikosti 12 B.

EMule po startu a výběru serveru, ke kterému se má připojit, zasílá klient mnoho paketů ‚*požadavek na stav serveru*' a ‚*zjisti informaci o serveru*'. Po připojení k síti Kad se v datovém toku objeví pakety o velikosti 27 B a 35 B.

Shareaza průběžně zasílá 19 B velké pakety.

Tímto zajímavým objevem se tedy dokonce dá zjistit přímo používaný klient. Samozřejmě by byla pro přesnější údaje třeba důkladnější analýza, pro názornost je to však postačující – P2P síť detekována byla.

3.2.2 Detekce porovnáváním tvaru křivky datového toku

Použijeme-li pro zobrazení datového toku nějaký vizualizační nástroj, můžeme si povšimnout, že charakteristika výsledné křivky bývá podobná. Této podobnosti by se dalo využít pro detekci konkrétního datového toku.

K řešení lze přistupovat vícero způsoby. Můžeme porovnávat buď jen počet příchozích nebo odchozích bytů a nebo oboje v součtu dohromady. Výsledné hodnoty vytvoří v časové závislosti nějakou křivku. Vytvoříme-li si křivky typického P2P provozu, můžeme poté s vytvořenou křivkou tyto vzorové křivky porovnávat. Shodují-li se nebo se jim velmi blíží, můžeme říct, že jsme detekovali P2P provoz.

Pro porovnání křivek či trajektorií existují různé algoritmy [11]. Ukažme si příklad vyhledání podobné křivky **rozsahovým dotazem**:
Uvažujeme *diskrétní* trajektorii (data nepřichází spojitě, ale po časových úsecích), trajektorie je tedy n-rozměrný vektor. Podobnost trajektorií je tří druhů: *prostorově-časová* (v $\mathbb{R}^n \times \mathbb{N}$), *prostorová* ($\mathbb{R}^n$) a *časová* ($\mathbb{R} \times \mathbb{N}$). Euklidovská vzdálenost L_2 mezi vektory u, v je:

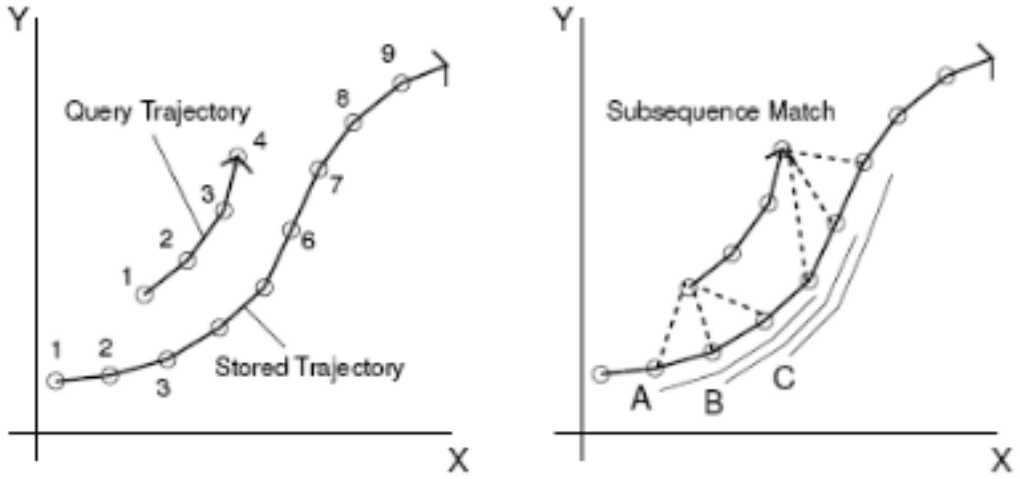
$$\text{Časová řada:} \quad \rho_{\mathbb{R}\times\mathbb{T}}(u,v) = \sqrt{((u-v)^2 + \ldots + (u_m - v_m)^2)}$$

$$\text{Zobecnění pro vyšší dimenze:} \quad \rho_{\mathbb{R}^n\times\mathbb{T}}(u,v) = \sqrt{(\rho_{\mathbb{R}\times\mathbb{T}}(u-v)^2 + \ldots + \rho_{\mathbb{R}\times\mathbb{T}}(u_m - v_m)^2)}$$

Rozsahový dotaz bere jako vstup posloupnost hodnot a jako výstup dává množinu všech částí uložených trajektorií, které splňují kritérium podobnosti. Zjednodušené hledání vypadá takto:

1. Vyberme jednu uloženou trajektorii
2. Prozkoumáme všechny její podsekvence na podobnost
3. Postup opakujeme pro všechny uložené trajektorie

Názorně jde princip vidět na obrázku 3.1. Máme-li množinu všech částí uložených trajektorií, můžeme říci, zda jde skutečně o P2P provoz či nikoliv. To záleží na tom, od jaké míry podobnosti prohlásíme, že se jedná o P2P provoz.



Obrázek 3.1: Hledání „nejpodobnější“ podsekvence délky shodné s vstupní posloupností, převzato z [11]

Nevýhodou tohoto řešení je použití v případě, kdy P2P provoz netvoří dominantní datový tok. Charakteristika a tvar křivky by neodpovídaly datovému toku P2P a výsledek by tedy mohl být nepřesný nebo matoucí.

3.2.3 Detekce neuronovými sítěmi

Zajímavou alternativou je detekce neuronovými sítěmi. Příklad implementace neuronové sítě pro klasifikaci datového přenosu byl popsán v diplomové práci O. Plchota [15]; obdobného způsobu by se dalo využít i u P2P sítí. Princip spočívá v naučení neuronové sítě (nastavení vah jednotlivých neuronů) na konkrétní P2P síť nebo množinu P2P sítí. Do naučené neuronové sítě je poté poslán datový tok a ta se pokusí vyhodnotit, zda jde o P2P provoz (klasifikace P2P provozu nebo jen ano/ne na otázku, zda datový tok obsahuje P2P provoz).

Možností, co konkrétně neuronovou sítí detekovat, je jistě vícero. Při použití způsobu popsaném v [15] však nemusíme dosáhnout, co jsme zamýšleli. Problém P2P provozu může být v tom, že jde o velké množství jednotlivých datových toků (TCP i UDP). Klasifikace podle tvaru takového toku tak nemá smysl, jelikož může jít o velice krátké datové toky,

které samy o sobě vůbec nemusí vypadat jako datový tok P2P. Bylo by možné skládat jednotlivé datové toky do jednoho velkého datového toku a dělat analýzu nad ním, šlo by však nejspíše o výpočetně velmi náročné řešení. Pro detekci neuronovými sítěmi platí to samé – pokud není P2P provoz dominantní, nemusí ho být snadné detekovat; například Bittorrent otevírá spousty portů, na kterých vůbec nemusí proběhnout datový přenos. Je však možné zaměřit se i na jiné vlastnosti datového přenosu, než je přímo datový tok, a to třeba na výše zmíněné otevřené porty.

Kapitola 4

Implementace statistické detekce sítí P2P

V této diplomové práci se v implementační části budu zabývat detekcí P2P provozu na základě modelu chování heuristickou analýzou pomocí neuronové sítě.

4.1 Návrh řešení

Za operační systém, na kterém bude aplikace vyvíjena, byl vybrán **GNU/Linux**. Programovacím jazykem bude **C/C++** z důvodu efektivity; tyto jazyky mají blízko k nízkoúrovňovému programování.
Pro usnadnění práce budou použity i knihovny třetích stran. Jednou z nich je **libpcap**[1], což je knihovna pro zachytávání síťových paketů na uživatelské úrovni. Touto knihovnou budou sbírana data. Další knihovnou bude **Fast Artificial Neural Network Library**[2] pro implementaci neuronové sítě.

Výsledkem bude aplikace určená pro příkazovou řádku (neinteraktivní), která bude pracovat ve dvou režimech – učícím a detekujícím. V učícím režimu se bude učit určitý datový tok P2P sítě. Čím více tohoto datového toku bude, tím více bude aplikace, respektive její neuronová síť, naučena. V detekujicím režimu bude aplikace kontrolovat tok dat a podle naučenosti se bude snažit detekovat, zda tento dat obsahuje P2P provoz. Aplikace skončí buď odpovědí *ano/ne* na otázku, zda datový tok obsahuje P2P síť, a nebo bude vypisovat informace týkající se datového toku a pravděpodobnosti výskytu P2P sítě (ideálně v nějakém snadno parsovatelném formátu pro případný grafický frontend).
Princip detekce bude následující: jak již bylo zmíněno v sekci 3.2.3, není výhodné dělat detekci nad jednotlivými datovými toky, jelikož by to bylo výpočetně náročné. Možnou variantou je sledování využití jednotlivých portů TCP a UDP. Těchto portů je 65536, pro TCP a UDP je to tedy $65536 \times 2 = 131072$ portů. Vytvoříme-li namapování port–datová struktura, kde datová struktura by obsahovala počet přijatých či odeslaných bytů a její velikost se bude pohybovat řádově v několika bytech (řekněme 16 B), velikost výsledné mapy portů by byla v řádu jednotek megabytů, což je pro dnešní počítačové systémy unesitelné (v případě 16 B velkých struktur by šlo o 2 MB velkou mapu). Tuto mapu by využila neuronová síť (2×65536 či 131072 vstupních neuronů).

[1] http://sourceforge.net/projects/libpcap/
[2] http://leenissen.dk/fann/

Kapitola 5

Závěr

Byly rozebrány různé typy P2P sítí a provedena jejich klasifikace. Také bylo popsáno, co je pro jednotlivé sítě typické, co se datového toku týče. Jako příklad byly probrány detailněji protokoly 3. P2P sítí: Direct Connect, Bittorrent a Skype.

Poté byly popsány způsoby detekce P2P sítí. Detekce se dá rozdělit na 2 hlavní směry: detekce založená na informacích z hlaviček paketů (detekce podle portu nebo analýzou protokolu na aplikační vrstvě ISO/OSI) a detekce založená na chování datového toku.

Detekce založená na informacích z hlaviček paketů se ukazuje v poslední době jako ne příliš výkonna, protože tvůrci P2P sítí používají techniky, které tento typ detekce komplikují – používání náhodných portů, tunelování, šifrování atd.

Detekce založená na chování datového toku je sofistikovanějším způsobem, jak zjistit, zda je na síti P2P provoz. Byly předloženy tři způsoby detekce:
Detekce sledováním UDP paketů patří k nejjednodušším způsobům. Ukazuje, že P2P aplikace vykazují oproti běžnému provozu větší využití protokolu UDP a pro P2P protokoly je typické určité chování zasílání UDP paketů. Na základě tohoto je možné detekovat P2P provoz a někdy i přímo používaného klienta.
Detekce porovnáním křivky datového toku se snaží porovnávat graf datového provozu se známými tvary křivek datových toků P2P. Výhodou je přesnost v případě dobrých porovnávacích vzorků a vhodného porovnávacího algoritmu.
Třetím uvedeným způsobem je detekce pomocí neuronových sítí. Tato technika se dá aplikovat na více vlastností datových toků; v této práci byla zvolena detekce podle využití portů TCP a UDP. Některé porty mohou být otevřeny, aniž by na nich byl odeslán jediný byte informací, což se neprojeví na celkovém datovém toku a ostatní metody by zde mohly selhat.

Navrhovaný a implementovaný způsob řešení detekce P2P provozu neuronovými sítěmi by mohl být vhodným nástrojem pro administrátory, kteří potřebují z nějakého důvodu P2P provoz detekovat. Protože jsou P2P sítě a protokoly stále sofistikovanější, bude třeba i pokročilejčích metod detekce, mezi které patří právě zmiňovaná detekce neuronovými sítěmi.

Literatura

[1] Heterogeneous sensor networks. http://www.intel.com/research/exploratory/heterogeneous.htm [online]. Poslední modifikace 6.1. 2008.

[2] Open direct connect – documentation. http://sourceforge.net/docman/?group%20id=36589 [online]. Poslední modifikace 5.1. 2008.

[3] Zákon o právu autorském, o právech souvisejících s právem autorským a o změně některých zákonů (autorský zákon). http://www.mvcr.cz/sbirka/2000/sb036-00.pdf [online], duben 2000. Poslední modifikace 5.1. 2008.

[4] Protocol specifications v1.0. http://www.bittorrent.org/protocol.html [online], 2007. Poslední modifikace 25.12. 2007.

[5] Stephanos Androutsellis-Theotokis a Diomidis Spinellis. A survey of peer-to-peer content distribution technologies. *ACM Comput. Surv.*, 36(4):335–371, 2004.

[6] Dostálek Libor a Kabelová Alena. *Velký průvodce protokoly TCP/IP a systémem DNS*. Computer Press, 3. edition, 2002.

[7] Thomas Karagiannis a Konstantina Papagiannaki a Michalis Faloutsos. Blinc: multilevel traffic classification in the dark. http://www.sigcomm.org/sigcomm2005/paper-KarPap.pdf [online], 2005. Poslední modifikace 26.12. 2007.

[8] Carmen Carmack. How bittorent works. http://computer.howstuffworks.com/bittorrent2.htm [online]. Poslední modifikace 6.1. 2008.

[9] François Deppieraz. Peer-to-peer detection. Master's thesis, Haute Ecole d'Ingéniere et de Gestion du Canton de Vaud, 2006.

[10] Yiming Gong. Identifying p2p users using traffic analysis. http://www.securityfocus.com/infocus/1843/1 [online], 21.7. 2005. Poslední modifikace 25.12. 2007.

[11] Jiří Jakl. Podobnost trajektorií. http://siret.ms.mff.cuni.cz/skopal/mdb/referaty/2006/Trajektorie.ppt [online], 2006. Poslední modifikace 26.12. 2007.

[12] Thomas Karagiannis, Andre Broido, Michalis Faloutsos, and Kc claffy. Transport layer identification of p2p traffic. citeseer.ist.psu.edu/karagiannis04transport.html [online], 2004.

[13] Vojtěch Kment. Skype – telefonování zadarmo. http://www.lupa.cz/clanky/skype-8211-telefonovani-zadarmo/ [online], červenec 2006. Poslední modifikace 6.1. 2008.

[14] Jiří Peterka. Počítačové sítě jako důsledek vývoje. http://www.earchiv.cz/a94/a412c500.php3 [online], 1994. Poslední modifikace 6.1. 2008.

[15] Oldřich Plchot. Systém pro uživatelem řízené QoS. Master's thesis, FIT VUT v Brně, 2007.

[16] Todd Sundsted. The practice of peer-to-peer computing: Introduction and history. http://www.ibm.com/developerworks/java/library/j-p2p/ [online], 1.3. 2001. Poslední modifikace 24.12. 2007.

[17] více autorů. Peer to peer (p2p) introduction and history. http://www.mac-p2p.com/p2p-history/ [online]. Poslední modifikace 24.12. 2007.

[18] více autorů. Bittorrent protocol specification v1.0 – notes. http://wiki.theory.org/index.php/BitTorrentSpecification#Notes [online], 22.12. 2007. Poslední modifikace 25.12. 2007.

[19] více autorů. Direct connect (file sharing). http://en.wikipedia.org/wiki/Direct_connect_file-sharing_application [online], 23.12. 2007. Poslední modifikace 24.12. 2007.

[20] více autorů. Peer-to-peer. http://cs.wikipedia.org/wiki/Peer-to-peer [online], 5.10. 2007. Poslední modifikace 24.12. 2007.

[21] více autorů. Peer-to-peer. http://en.wikipedia.org/wiki/Peer-to-peer [online], 23.12. 2007. Poslední modifikace 24.12. 2007.

[22] více autorů. Skype protocol. http://en.wikipedia.org/wiki/Skype_Protocol [online], 10.12. 2007. Poslední modifikace 24.12. 2007.