VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY

FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE A ANALÝZA PROVOZU P2P SÍTĚ BITTORRENT SYNC

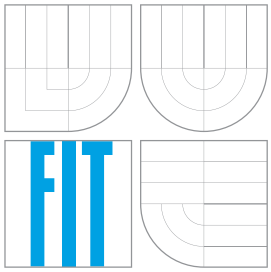
BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KUTLÁK

BRNO 2015

VYSOKÉ UČENÍ TECHNICKÉ V BRNĚ
BRNO UNIVERSITY OF TECHNOLOGY



FAKULTA INFORMAČNÍCH TECHNOLOGIÍ
ÚSTAV INFORMAČNÍCH SYSTÉMŮ

FACULTY OF INFORMATION TECHNOLOGY
DEPARTMENT OF INFORMATION SYSTEMS

DETEKCE A ANALÝZA PROVOZU P2P SÍTĚ BITTORRENT SYNC

DETECTION AND ANALYSIS OF P2P NETWORK BITTORRENT SYNC

BAKALÁŘSKÁ PRÁCE
BACHELOR'S THESIS

AUTOR PRÁCE
AUTHOR

MARTIN KUTLÁK

VEDOUCÍ PRÁCE
SUPERVISOR

Ing. PETR MATOUŠEK, Ph.D.

BRNO 2015

Abstrakt

Bakalářská práce se věnuje problematice detekce P2P sítě BitTorrent Sync v síťovém provozu. V teoretické části práce je představena architektura P2P sítě, protokol BitTorrent Sync a jsou zde uvedeny možné metody pro detekci P2P komunikace. Jsou zde také uvedeny výsledky analýzy jednotlivých typů přenosů a fází komunikace aplikace BitTorrent Sync. Na základě těchto znalostí je navrhnut nástroj pro monitorování komunikace BitTorrent Sync. V praktické části práce je poté představena implementace tohoto nástroje a jsou zde prezentovány výsledky provedených testů.

Abstract

Bachelor's thesis is focused on issues with detection of the P2P network BitTorrent Sync in network traffic. In the theoretical part of this work is introduced architecture of the P2P networks, BitTorrent Sync protocol and possible methods for P2P network detection. On the base of this knowledge is designed tool for detection of BitTorrent Sync network traffic. In the practical part of this thesis is presented implementation of the whole monitoring system. In the end of practical part are presented results of conducted tests with monitoring system.

Klíčová slova

BitTorrent, P2P, Sync, detekce, analýza, monitorování, síťový provoz

Keywords

BitTorrent, P2P, Sync, detection, analysis, monitoring, network traffic

Citace

Martin Kutlák: Detekce a analýza provozu P2P sítě BitTorrent Sync, bakalářská práce, Brno, FIT VUT v Brně, 2015

Detekce a analýza provozu P2P sítě BitTorrent Sync

Prohlášení

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

.....
Martin Kutlák
21. května 2015

Poděkování

Tímto bych chtěl poděkovat panu Ing. Petru Matouškovi, Ph.D. za jeho cenné rady a odbornou pomoc při tvorbě této bakalářské práce.

Obsah

Kapitola 1

Úvod

Peer-to-peer (P2P) sítě jsou v dnešní době velmi populární a jejich využití v síťových aplikacích neustále roste. Principu P2P se nejvíce využívá v aplikacích pro sdílení souborů, ale jejich uplatnění nalézt i u hlasových služeb nebo vysílání multimediálního obsahu. A právě kvůli této rostoucí popularitě a charakteristice P2P sítí, kterou je vytváření velkého množství spojení se zasíláním malých paketů mající za následek zabrání velké části přenosového pásma, je potřeba tuto komunikaci umět detekovat. Dalším důvodem pro detekci P2P aplikací může být ochrana proti úniku důležitých dat. Pomocí detekce tak poté můžeme vytvářet statistiky o využití sítě, či klasifikovat provoz na síti.

Jednou z aplikací, která využívá P2P sítě je BitTorrent Sync. Tato aplikace slouží jako nástroj pro synchronizaci souborů nebo jako uživatelské úložiště. Dokáže synchronizovat soubory mezi zařízeními jak v lokální síti, tak i mezi vzdálenými zařízeními pomocí P2P. Jedná se tak o alternativu služeb jakou jsou Dropbox, Google Drive nebo OneDrive. Výhodou BitTorrent Sync je, že soubory nejsou nikdy ukládány na vzdálený server, nýbrž jsou zasílány pouze mezi uživateli. Pro sdílení souborů jsou v aplikaci generovány klíče[1], které uživatel může zaslat jiným uživatelům. Tímto tak má kontrolu nad tím, kdo bude mít k souborům přístup.

Předmětem této práce je provedení analýzy komunikace aplikace BitTorrent Sync, při níž bychom měli zjistit, s kým klient komunikuje, jaké druhy komunikací se používají a jaké typy zpráv se při těchto komunikacích používají. Pomocí těchto zjištěných poznatků je poté vytvořen nástroj pro detekci takové komunikace pro účely monitorování. Nástroj by měl také poskytovat vhodné zobrazení výsledků.

Ve druhé kapitole je popsáno fungování architektury peer-to-peer, protokolu BitTorrent a v poslední části je detailněji popsána samotná aplikace BitTorrent Sync a její protokol. Ve třetí kapitole jsou představeny obecné možnosti detekce peer-to-peer komunikace a metody, které se pro detekci používají. Ve čtvrté kapitole je proveden rozbor komunikace. Jsou zde uvedeny metody, které klient používá pro objevování nových uzlů, typy komunikací, které se používají pro komunikaci mezi uzly a ukázky dat, které jsou posílány. V páté je uveden návrh nástroje pro detekci, který vychází z poznatků zjištěných ve čtvrté kapitole. Dále je v této části popsána implementace nástroje a jsou zde popsány výstupy nástroje. Kapitola šest obsahuje popis testování nástroje a výsledky testů. V samotném závěru práce jsou zhodnoceny dosažené výsledky práce, možné použití nástroje v praxi a jsou zde uvedeny možnosti, jak lze práci dále rozvíjet.

[1] Klíč je unikátní identifikátor náhodně generovaný pro sdílenou složku.

Kapitola 2

Sítě peer-to-peer

Pro pochopení toho, jak služba BitTorrent Sync funguje, je důležité porozumět technologiím, které ke svému fungování využívá. Proto bude v této kapitole nejdříve popsána architektura peer-to-peer sítě, dále pak bude představen protokol BitTorrent, který využívá stejných prostředků jako služba BitTorrent Sync. V poslední částí této kapitoly bude popsána služba BitTorrent Sync.

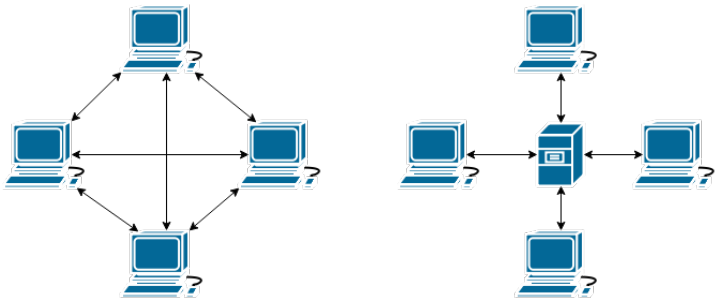
2.1 Architektura P2P sítí

V sítích typu klient-server existují dva základní prvky - server, který se poskytuje různé služby a klient, který od serveru služby žádá. Komunikace mezi nimi probíhá způsobem dotaz-odpověď. Klient zasílá požadavky na služby na server, který tyto požadavky zpracuje a odešle klientovi odpověď. Nevýhodou tohoto přístupu je, že server je při stoupajícím počtu požadavků od klientů více a více zatěžován [10].

V případě architektury sítě P2P prvky klient a server nerozlišujeme. V P2P sítí jsou si všechny uzly (anglicky peer) navzájem rovny. Při komunikaci mezi sebou tak každý uzel zajišťuje zároveň serverové i klientské služby. Toto ovšem přináší i rizika, například při správě a zabezpečení dat přenášených v síti. Spolu komunikující uzly se stávají více náchylné vůči různým bezpečnostním hrozbám [10]. Porovnání architektur klient-server a P2P lze vidět na obr. 2.1.

Komunikaci dvou uzlů můžeme rozdělit do těchto fází:

- Signalizace – Zahrnuje vytvoření spojení a vyhledávání. Během této fáze uzel zjišťuje, na kterých uzlech se nacházejí požadovaná data.
- Datový přenos – V této fázi dochází ke kontaktování a zahájení stahování požadovaných dat od konkrétního nalezeného uzlu [6].



Obrázek 2.1: Porovnání architektur P2P(vlevo) a klient server (vpravo).

2.2 Protokol BitTorrent

BitTorrent je protokol navržený k distribuci souborů mezi velkým množstvím uzlů s minimálním zatížením na původní zdroj těchto souborů [9]. Toho je dosaženo tak, že soubor je rozdělen na několik menších bloků, které jsou vzájemně distribuovány ve skupině uzlů. Tato skupina uzlů se nazývá roj (swarm) [1].

Uzel může být členem několika rojů a sdílet tak několik souborů zároveň. Před zahájením stahování dat musí uživatel nejdříve stáhnout soubor .torrent s metadaty z některých z indexujících webových stránek. Klient aplikace BitTorrent poté interpretuje metadata a zjištěné informace využije k nalezení aktivních uzlů za pomoci některé z těchto metod:

- Synchronizační server – Synchronizační servery jsou servery starající se o seznam s uzly sdílející data (anglicky seeders, jsou to uzly, které mají dostupný kompletní soubor a pouze ho sdílejí) a s uzly, kteří data stahují (anglicky leechers, uzly mající nekompletní kopii souboru). Během přenosu dat klient aplikace odesílá zprávy se svým stavem na synchronizační server. Takto dochází k aktualizaci údajů seznamu s aktivními uzly

- Tabulka DHT (Distributed Hash Table) – Metoda umožňující nacházet informace o uzlech dotazování se jiných BitTorrent klientů bez nutnosti použití synchronizačního serveru. Každý záznam o uzlu v tabulce DHT je spojen s rojem, ve kterém je uzel aktivní. BitTorrent a BitTorrent Sync využívají tabulku DHT založenou na protokolu Kademila, která pro komunikaci využívá protokol UDP.

- Výměna uzlů – Jedná se o metodu během níž si dva komunikující uzly vymění informace o dalších uzlech, které jsou součástí roje [3].

Jakékoliv řídící požadavky, odpovědi a metadata jsou přenášeny v zakódované podobně, které se nazývá bencoding. Takto zakódovaná data se skládají ze seznamů a asociativních polí obsahující páry klíč:hodnota. Ke každému klíči a k němu odpovídající hodnotě předchází informace o jejich délce (byte) a dvojtečka. Pro příklad odpověď se seznamem uzlů by vypadala takto *1:m5:peers* [2]. Jednička v tomto příkladu znamená, že bude následovat jeden znak. Tímto znakem je m, což znamená, že bude následovat metoda zprávy. Pětka nám říká, že délka metody zprávy bude 5 znaků dlouhá. *Peers* je pak typ metody, který příjemci oznamuje, že je mu zasílán seznam uzlů pro dané ShareID.

2.3 Aplikace BitTorrent Sync

BitTorrent Sync (zkráceně BTSync) je aplikace určená k synchronizaci dat. Aplikace byla vytvořena firmou BitTorrent Inc. v roce 2013. Jejím hlavním cílem je synchronizovat obsah vybrané složky mezi dvěma a více zařízeními. Při jakýchkoliv změnách ve sdílené složce na jednom zařízení jsou pak tyto změny rozeslány i na ostatní zařízení. Může tedy sloužit i jako náhrada za webová úložiště bez nutnosti použití webového serveru, na který by se data ukládala [13]. Z absence webového serveru potom plyne nevýhoda, že zařízení, se kterým chceme data sdílet, musí být online.

Mezi výhody BTSync patří úložná kapacita (na rozdíl od webových služeb je omezená pouze velikostí místa na disku) nebo soukromí (data jsou uložena pouze lokálně na zařízeních a komunikace mezi zařízeními je šifrována pomocí algoritmu AES-128 [11]).

Pro přenos dat používá BTSync transportní protokol TCP a protokol μTP (Micro Transport Protocol). Tento protokol nahradil v roce 2009 BitTorrent protokol, který pro přenosy dat využíval TCP. μTP pro přenosy využívá transportního protokolu UDP. Díky tomu může μTP plně využít nepoužívané přenosové šířky pásma, bez narušení ostatních internetových připojení [7].

K připojení zařízení mezi sebou BTSync využívají klíče. Klíče se mimo jiné používají k identifikaci složek, k zakódování dat nebo k vyhledávání uzlů v síti internet. Klíče jsou řetězce (20 byte), které jsou náhodně generovány aplikací BTSync. Existuje několik druhů klíčů, a to, o jaký druh se jedná, se pozná podle prvního písmene klíče. Může jít například o klíč s plným přístupem (uzel s tímto klíčem má práva číst i zapisovat data), začínající písmenem A, klíč s omezeným přístupem (uzel s tímto klíčem může data pouze číst), začínající písmenem B, a nebo klíč na jedno použití (klíč s omezený časově nebo počtem přístupů) začínající písmenem C [12].

Příklady klíčů:

- AEBU4RVOZ6LJQQFDWQMK5W7LQKKDW3RSB – klíč s plným přístupem
- BJYC4QQOH73VY5DHRNFKQ7DPPKMB3TI6K – klíč pouze pro čtení dat
- CAIWMYLCMI6DZJBBBRMB7D6U2EZNQN5XS – klíč pro jedno použití a pouze pro čtení dat

Při generování klíče pro čtení a zápis (RW klíč) se automaticky také generuje [12]

- Pár asymetrických klíčů odvozených od RW klíče, které se využívají k podepisování a ověřování hashe souborů.
- Klíč pouze pro čtení (RO klíč), odvozený od veřejného klíče.
- ShareID odvozené od RO klíče.
- Klíč pro šifrování přenosu, který je generován zvlášť pro každé spojení. Tento klíč je odvozen od RO klíče.

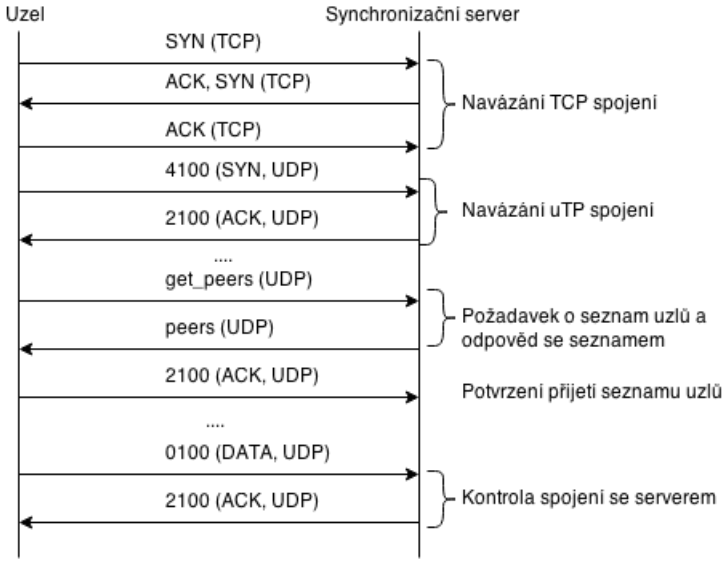
Po získání nebo vygenerování klíče se BTSync pokusí najít uzly se stejným klíčem jednou z těchto metod:

- Synchronizační server – Klient aplikace BTSync se připojí na synchronizační server[1] a zašle mu dotaz na seznam uzlů se stejným klíčem. V dotazu je obsažena i interní IP

[1] Adresy synchronizačních serverů jsou uloženy v konfiguračním souboru, který klient stahuje z adresy config.usyncapp.com

adresa a port uzlu. Synchronizační server uzlu odpoví informacemi (IP adresy a porty) o uzlech. Po získání těchto informací se uzel pokusí připojit k ostatním uzlům.

- Tabulka DHT – Aplikace BTSync nejdříve pomocí DNS[2] dotazů zjistí IP adresy serverů `router.bittorrent.com` a `router.utorrent.com`. Na tyto zjištěné IP adresy a port `6881` pak zašle požadavek pro získání tabulky s uzly. Po získání tabulky začne uzlům v této tabulce zasílat dotazy, zda nevlastní ShareID sdílené složky. Pokud uzel toto ShareID nevlastní odešle jako odpověď jeho hashovací tabulku s uzly. Jinak se uzly k sobě pokusí připojit.
- Vyhledání v lokální síti – K vyhledávání se využívá multicast. Při vyhledávání uzlů v lokální síti klient aplikace zašle požadavek k připojení do multicastové skupiny `239.192.0.0`. Pro vyhledávání uzlů se pak odesílají pakety se ShareID sdílené složky a IP adresa a port na kterou má být zaslána odpověď. Pokud uzel příjme paket se ShareID které vlastní, pokusí se připojit k odesílateli tohoto paketu.
- Známé uzly – Zadáním známých uzlů (IP adresy a portu) v nastavení sdílení složky se uzel připojí ke známým uzlům přímo, bez využití předchozích metod [12].



Obrázek 2.2: Ukázka možné komunikace uzlu se synchronizačním serverem.

Na obr. 2.2 lze vidět ukázku možné komunikace uzlu se synchronizačním serverem. Komunikace začíná navázáním TCP spojením se synchronizačním serverem. Dále pak následuje navázání μTP spojení. To na rozdíl od navazování TCP spojení probíhá pouze ve dvou krocích. Po navázání spojení následuje zaslání požadavku na seznam uzlů. Tyto zprávy mohou být zasílány jak pomocí TCP, tak i μTP (ten běží na protokolu UDP).

2.4 Protokol aplikace BitTorrent Sync

Jak již bylo zmíněno v 2.3, aplikace BTSync pro komunikaci využívá obou transportních protokolů, TCP i UDP. Volba transportního protokolu, který je použit při přenosu se určuje

[2]DNS – služba pro překlad adres

podle vytížení přenosového pásma. Pokud by použití TCP omezilo ostatní aplikace, jako jsou například VoIP nebo webový prohlížeč, použije se místo něj UDP.

Pro komunikaci mezi uzly nebo mezi uzly a servery se používá několika druhů zpráv. Tyto zprávy mají jasně danou strukturu a každá slouží k jinému účelu. V následujícím textu proto bude popsána jejich struktura a budou uvedeny příklady, kdy se tyto zprávy používají.

2.4.1 Zprávy pro vyhledávání uzlů

Prvním druhem zpráv, které se při komunikaci používají jsou zprávy obsahující metodu *ping*[3]. Při komunikaci se tento druh zpráv používá pro vyhledávání uzlů vlastnící stejné ShareID. Struktura zasílaného paketu vypadá následovně.

BSYNC	Hlavička BTSync
00	Nulová hodnota
d	Označení začátku asociativního pole
1:m	Označení identifikátor metody zprávy
4:ping	Metoda zprávy
4:peer	Označení identifikátoru pro PeerID
20:PeerID	PeerID uzlu, který zprávu zaslal
5:share	Označení identifikátoru pro ShareID
20:ShareID	ShareID sdílené složky
e	Konec asociativního pole

Tabulka 2.1: Tabulka s ukázkou dat obsažených v paketu zasílaném ostatním uzlům.

Hlavičku paketu tvoří řetězec *BSYNC* následovaný nulovou hodnotou. Po úvodní hlavičce následuje asociativní pole. Začátek tohoto pole je značen znakem *d*. Dále pak následují dvojice klíč:hodnota. Před každým klíčem a hodnotou je uvedena jejich délka. První hodnotou nacházející se v poli je informace o metodě zprávy. V tomto případě *ping*. Další hodnotou je PeerID odesílatele paketu (109E9EBCBABC4DA60828E5A8A8BCDBF9DD042150), následně se zde může vyskytovat číslo portu, na který má být odeslána odpověď. Číslo portu se vkládá pouze do paketů rozesílané po multicastu, kde nelze určit na kterém portu odesílatele paketu běží aplikace BTSync. Poslední informací, která se v těchto paketech vyskytuje je hodnota ShareID (B9E20BBB7AC84BD2F2C5AACE43D6FE5395AA07F1). Poté již následuje konec asociativního pole značený znakem *e*. Tímto znakem je paket ukončen.

2.4.2 Obecné zprávy

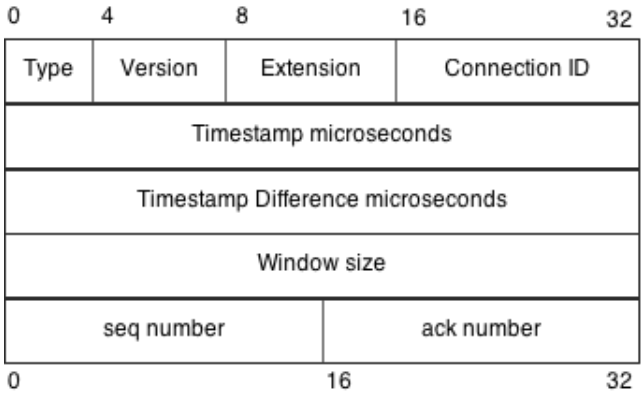
Obecné zprávy slouží k informování komunikujících uzlů o spojení nebo také k přenosům dat mezi uzly. Využívají se k navazování spojení s uzly nebo se synchronizačním serverem. Jsou posílány pomocí transportního protokolu UDP. Na začátku datové části těchto paketů se nachází μTP hlavička, obr. 2.4.2.

Hlavička je složena ze 4 bitů určující typ paketu, 4 bitů určující verzi protokolu (v současnosti verze 1). Dále pak 8 bitů pro možná rozšíření, 16 bitů pro číslo identifikující všechny pakety patřící stejnému spojení. Uzel zahajující spojení zvolí, které číslo bude použito, spojení pro odpověď má pak toto číslo o jedničku větší.Následujících 32 bitů slouží pro uložení

[3]S nástrojem ping, který se používá k zjištění dostupnosti vzdáleného síťového rozhraní, sdílí pouze jméno.

času odeslání paketu, dalších 32 bitů pak pro uložení rozdílu času mezi přijetím a odesláním paketu. Poté následuje 32 bitů oznamující šířku okna, 16 bitů pro sekvenční číslo, určující pořadí pro zpracování paketů a 16 bitové sekvenční číslo, posledně přijatého paketu [7]. Zprávy tedy můžeme rozdělit podle typu v μTP hlavičce. Těmi jsou:

- 01 – Zpráva pro přenos dat.
- 11 – Zpráva odpovídá příznaku FIN. Tato zpráva se používá k ukončení spojení mezi dvěma uzly nebo mezi uzlem a synchronizačním serverem.
- 21 – Zpráva odpovídající příznaku ACK. Při komunikaci se používá k potvrzení přijetí dat, potvrzení navázání spojení, aj.
- 31 – Zpráva odpovídající příznaku RST. Používá se k resetování spojení.
- 41 – Zpráva odpovídající příznaku SYN. Využívá se jako požadavek k navázání spojení.



Obrázek 2.3: Hlavička protokolu μTP.

2.4.3 Komunikace se synchronizačním serverem

Ke komunikaci se synchronizačním serverem lze využít obou transportních protokolů (TCP i UDP). Transportní protokol TCP se však používá v komunikaci se synchronizačním serverem pouze pro stažení seznamu uzlů. Jinak se pro komunikaci se synchronizačním serverem používá protokolu μTP.

Během komunikace se synchronizačním serverem se posílají zprávy uvedené v části 2.4.2. Pro navázání spojení se synchronizačním serverem se používá dvojcestný handshake. Princip je následující. Uzel pošle zprávu s μTP typem SYN, na který mu server vrací odpověď s typem ACK. Po úvodním navázání spojení zasílá uzel požadavek (get_peers) na seznam uzlů pro ShareID uvedené v tomto paketu. Strukturu paketu s požadavkem lze vidět na tab. 2.2.

d	označení začátku asociativního pole
2:la	lokální adresa
6:	lokální IP adresa
2:lp	lokální adresa
4:	lokální port
1:m	označení identifikátor metody zprávy
9:get_peers	metoda zprávy, požadavek pro získání seznamu uzlů
4:peer	PeerID uzlu
20:PeerID	PeerID uzlu, který zprávu zaslal
5:share	označení identifikátoru pro ShareID
20:ShareID	ShareID sdílené složky
e	označení konce asociativního pole

Tabulka 2.2: Tabulka se strukturou paketu s metodou get_peers zasílaném na synchronizační server.

Na začátku paketu se nachází μTP hlavička popsaná v sekci 2.4.2. Po hlavičce následuje asociativní pole začínající znakem *d*. První položkou v poli je lokální IP adresa a port uzlu. Pokud se uzel nachází za systémem NAT je jeho lokální adresa a externí adresa odlišná. Externí IP adresu a externí port tak tedy server získává sám z IP hlavičky přijatého paketu. Další položkou je metoda zprávy, ta je v tomto případě *get_peers*. Posledními dvěma hodnotami v poli jsou pak PeerID odesílajícího uzlu a ShareID sdílené složky. Posledním znakem je znak *e*, kterým je ukončeno asociativní pole i celý paket.

Odpovědí na požadavek *get_peers* je zpráva s metodou *peers*. Ta obsahuje seznam uzlů pro ShareID obsažené v paketu s metodou get_peers. Strukturu tohoto paketu lze vidět na tab. 2.3.

d	označení začátku asociativního pole
2:ea	označení externí adresy příjemce zprávy
6:	externí IP adresa a port
1:m	označení identifikátor metody zprávy
5:peers	metoda zprávy, zpráva obsahuje seznam uzlů
5:peers	označení seznamu uzlů
ld	začátek seznamu uzlů
1:a	označení externí adresy a portu
6:	externí IP adresa a port uzlu v seznamu
2:la	označení lokální adresy
6:	lokální IP adresa
1:p	označení PeerID
20:	PeerID uzlu
e	konec seznamu uzlů
5:share	označení ShareID
20:	ShareID sdílené složky
4:time	časové razítko
e	označení konce asociativního pole

Tabulka 2.3: Tabulka s ukázkou struktury paketu odpovědi synchronizačního serveru na požadavek get_peers.

2.5 Shrnutí

Tato kapitola sloužila k představení architektury sítí P2P, protokolu BitTorrent a aplikace BTSync. V části o architektuře sítě P2P je vysvětleno fungování těchto sítí a rozdíl oproti architektuře klient-server. Následuj popis protokolu BitTorrent a metody, které se využívají během komunikace uzlů. V části o aplikaci BTSync je uvedeno, k čemu aplikace slouží a jakých prostředků využívá pro sdílení souborů mezi uzly. Je zde také uveden protokol aplikace BTSync. Znalosti získané v této kapitole jsou v další části práce využity při analýze komunikace BTSync.

Kapitola 3

Detekce P2P komunikace

Tato kapitola slouží pro představení metod pro detekci P2P komunikace. Představíme si tři přístupy, jak takovou komunikaci detekovat. První metoda je založená na mapování portů, další metoda analyzuje obsah paketů (*Deep Packet Inspection*), třetí metodou je analýza síťových toků (*Deep Flow Inspection*). Další možností, jak detekovat P2P komunikaci, je kombinace některých z předchozích metod. Každá z těchto metod má své výhody a nevýhody, které budou v následujícím textu objasněny.

3.1 Detekce podle čísel portů

Jedná se o nejstarší a nejjednodušší přístup k detekování provozu P2P aplikací v síťovém provozu. Tato metoda vychází z konceptu, že mnoho P2P aplikací využívá jednoho předdefinovaného portu, viz tab. 3.1. Při komunikaci s okolím pak používají právě tento port.

Při detekci komunikace založené na mapování portů se sleduje síťový provoz a kontroluje se, zda se některý ze známých čísel portů neshoduje. Při shodě těchto portů je komunikace označena za P2P komunikaci. Nevýhodou této metody ovšem je, že většina P2P aplikací dovoluje uživateli změnit číslo portu, na kterém probíhá komunikace, nebo jsou porty náhodně měněny při spuštění aplikace.

Přestože je tato metoda v dnešní době pro P2P sítě značně neefektivní, lze ji použít pro detekci protokolů s neměnným portem (HTTP, DNS, FTP) [4].

Aplikace	**TCP**	**UDP**
Limewire	6346/6347	6346/6347
Morpheus	6346/6347	6346/6347
BearShare	6346	6346
Edonkey	4662	–
EMule	4662	4672
Bittorrent	6881-6889	6881-6889
WinMx	6699	6257

Tabulka 3.1: Aplikace a jejich předdefinované porty

3.2 Detekce pomocí analýzy obsahu paketů

Metoda analýzy obsahu paketů (*Deep Packet Inspection*) je založená na vyhledávání signatur[1] v datové části paketů. P2P aplikace si vyměňují zprávy, které často dodržují jisté charakteristiky. Při zkoumání provozu P2P protokolů se vytvoří databáze signatur pro tyto protokoly. Vyhledáváním signatur z databáze v datové části aplikační vrstvy paketu lze potom přesně určit, o který P2P protokol se jedná.

Tato metoda patří k nejpoužívanějším a je také jedna z nejefektivnějších, má ovšem také několik nevýhod. Jednou z nich je, že data v datové části nesmí být šifrována, jinak se stává zcela nepoužitelnou. Další nevýhodou je možné porušování soukromí uživatelů, protože se prochází obsahem všech paketů. Signatury P2P protokolů se také mohou měnit a proto je potřeba neustále aktualizovat jejich databázi. Nevýhodou může být i potřebný velký výpočetní výkon.

3.3 Detekce pomocí analýzy síťových toků

Metoda analýzy síťových toků (*Deep Flow Inspection*) je mnohem rychlejší a má nižší požadavky na výpočetní výkon, ovšem je méně přesná než metoda DPI. Dokáže však identifikovat P2P provoz, bez nutnosti zkoumání obsahu paketů a nedochází tak k narušování uživatelova soukromí. Při detekci tak tedy nezáleží, jestli je komunikace šifrovaná nebo se jedná o novou neznámou P2P aplikaci. Tímto se odlišuje od metody DPI, která při detekci šifrované komunikace selhává.

Podstatou této metody je shromažďování statistik o tocích (flow statistics). Z těchto statistik se provádí identifikace provozu, který odpovídá popisu chování P2P aplikace. K identifikaci se používá klasifikátor, který je vytvořen podle statistických vlastností toků příslušné aplikace. Vlastnosti používané pro analýzu [5]:

- Čísla portů
- Zdrojové a cílové IP adresy
- UDP/TCP protokol
- Velikost paketů
- Počet přijatých paketů za sekundu
- Rychlosti přenosů
- Poměr úspěšných/neúspěšných spojení

Problémy mohou nastat pokud budeme chtít zobecnit popis chování na všechny P2P aplikace. Existují sice některé společné vlastnosti, ale každá aplikace se chová trochu jinak. Rozdíl totiž není pouze v chování aplikací, ale i v nastavení ze strany uživatele. Ten může v nastavení aplikace změnit porty,které budou použity, povolit šifrování přenosu dat nebo omezit počet spojení. Fungování aplikace se také může měnit s každou novou verzí.

Situaci však komplikuje přítomnost jiných aplikací založených na P2P architektuře (P2P TV, Skype), které způsobují problém s nesprávnými detekcemi (*false positives*) [10].

Společné vlastnosti P2P protokolů [5]:

[1] signatura – podpis, skupina charakteristických znaků pro daný protokol

- Počet navázaných spojení – Při komunikaci v P2P síti je typické navazování velkého počtu TCP spojení, případně UDP toků. To je zapříčiněno rozdělením stahovaného souboru na několik segmentů, které se pak stahují od různých uzlů.
- Obousměrná komunikace – Při sdílení souborů uživatel typicky část souboru stahuje, ale také zároveň část odesílá. Všechny P2P aplikace mají společné to, že se stahované množství dat podobá odesílanému množství dat. Z toho vyplývá, že se komunikuje oběma směry, případně převažuje počet odesílaných dat směrem od uživatele.
- Využívání obou protokolů UDP a TCP – Většina P2P aplikací využívá ke svému fungování oba transportní protokoly. Ke kontaktování ostatních uzlů se používá UDP, kdežto TCP se používá pro přenosy dat.
- Velký počet neúspěšně navázaných spojení – Tato vlastnost je důsledkem toho, že uživatelé své aplikace často zapínají a vypínají a nejsou jsou tak, na rozdíl od serverů, neustále dostupní. Informace o uzlu ovšem zůstávají ještě nějakou dobu aktivní. Při pokusu o kontaktovaní takového uzlu ale dojde k neúspěšnému spojení. Snaha o kontaktování nedostupného uzlu se však ještě několikrát zopakuje.
- Maximální velikost paketů – Pří sdílení souborů se P2P aplikace typicky snaží posílat co největší pakety. Hodnota maximální velikosti paketu se pohybuje okolo 1400 bajtů. Toto neplatí u P2P aplikací, které se snaží zabezpečit co nejnižší odezvu (např. přenos hlasu).
- Čísla portů nad 1024 – Dnešní P2P aplikace používají při spuštění náhodně zvolený port. Společnou vlastností těchto aplikací je, že se typicky jedná o číslo s hodnotou nad 1024. Výjimku tvoří P2P aplikace, které se snaží maskovat za jiné známé služby (HTTP, HTTPS).

Právě kvůli těmto společným vlastnostem může při pokusu o detekci komunikace konkrétní aplikace (např. BitTorrent) docházet k tomu, že budou označovány i pakety, které se této komunikace neúčastní.

3.4 Shrnutí

V této kapitole byly představeny metody, které lze použít pro detekci provozu P2P sítí. První metodou byla detekce podle čísel portů, která, jak se ukázalo, není v dnešní době pro detekci P2P sítí příliš efektivní a hodí se spíše pro detekci služeb s neměnným portem. Další představenou metodou byla detekce pomocí analýzy obsahu paketů. Tato metoda je i přes své nevýhody v dnešní době jedna z nejpoužívanějších a nejefektivnějších. Poslední představenou metodou byla detekce na základě analýzy síťových toků.

Kapitola 4

Analýza komunikace

V této kapitole se blíže podíváme na metody, které BTSync využívá k vyhledávání uzlů, které sdílejí stejný klíč. Za pomocí těchto informací poté budeme moci navrhnout systém, který bude provoz aplikace BTSync monitorovat.

4.1 Způsoby objevování uzlů

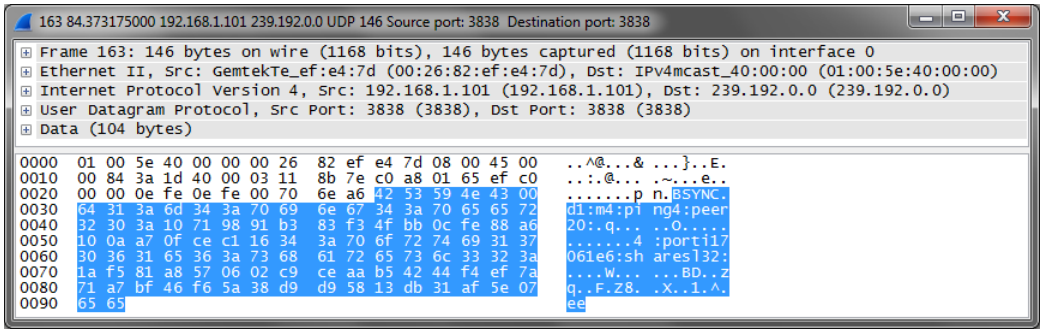
K vyhledávání nových uzlů využívá BTSync podobných principů jako BitTorrent. Tyto principy si v následující části projdeme a ukážeme si jak fungují. Každý ze způsobů vyhledávání nových uzlů lze v nastavení aplikace BTSync zapnout nebo vypnout. Ve výchozím nastavení je povoleno vyhledávání v lokální síti a použití synchronizačního serveru.

4.1.1 Vyhledávání v lokální síti

K nalezení nových uzlů v lokální síti využívá klientská aplikace multicastu. Po připojení do multicastové skupiny 239.192.0.0 začne klient do této skupiny rozesílat pakety. Tyto pakety obsahují hlavičku BSYNC, která sděluje, že se jedná o pakety aplikace BTSync. Dále pak následuje slovník zakódovaný v kódování bencoding. Tento slovník obsahuje informace o metodě, kterou je metoda *ping*[1], *PeerID*, port odesílatele na kterém běží aplikace BTSync a *ShareID* sdílených dat.

Uzly v síti připojení do stejné multicastové skupiny, kteří nesdílí stejné *ShareID*, které rozesílané pakety obsahují, tyto pakety ignorují. Pokud některý z uzlů v lokální síti sdílí stejné *ShareID*, odpoví zasláním paketu na IP adresu a port uvedený v přijatém paketu. Tato odpověď obsahuje pouze *PeerID* odpovídajícího klienta. Ukázku paketu zasílaného při vyhledávání v lokální síti lze vidět na obr. 4.1.

[1] V rámci komunikace aplikace BTSync se metoda nazvaná ping používá pro přenos různých informací.



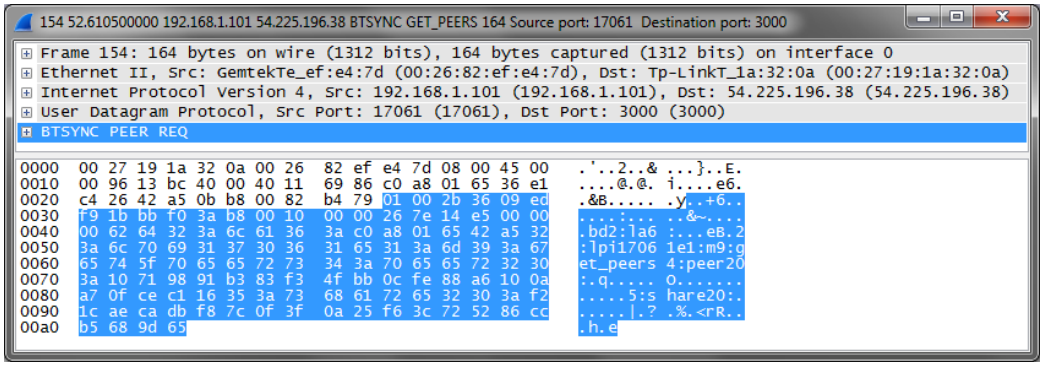
Obrázek 4.1: Multicastový paket pro vyhledávání uzlů v lokální síti.

4.1.2 Synchronizační server

Při použití této metody klient vyhledává nové uzly zasláním požadavků na synchronizační server, který se nachází na adrese t.usyncapp.com. Tato adresa se DNS dotazem překládá na jednu z těchto IP adres:

- 52.0.104.40
- 52.0.102.230
- 52.1.40.103
- 52.1.1.135

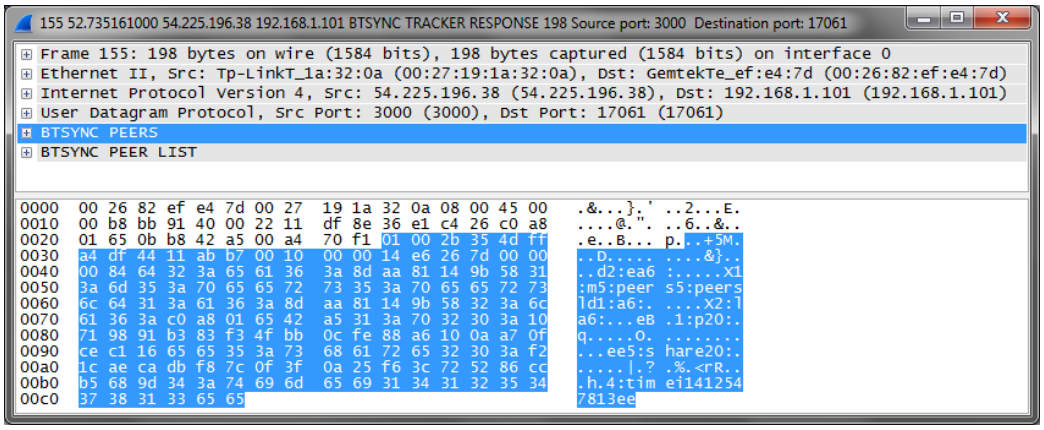
Tyto IP adresy získává klient z konfiguračního souboru. Prvním krokem je kontaktování synchronizačního serveru. Klient zahájí se serverem trojcestný handshake a zároveň pošle na server také UDP paket, jehož datová část obsahuje μTP hlavičku s typem zprávy *ST_ SYN*. Po navázání spojení zašle klient požadavek *get_peers* na synchronizační server (obr. 4.2). Tento požadavek je zaslán pomocí transportního protokolu TCP i protokolu UDP. Server po přijetí tohoto požadavku přidá IP adresu a port klienta, které jsou uvedené v obdrženém paketu, do seznamu aktivních uzlů pro *ShareID*, které je rovněž obsaženo v obdrženém paketu. Server poté odpoví zasláním seznamu uzlů se stejným *ShareID*. Seznam obsahuje IP adresy, porty a *PeerID* uzlů. Paket obsahuje také *ShareID* sdílené složky, na kterou se uzel dotazoval. Po obdržení seznamu s uzly se ukončí TCP spojení se serverem a další komunikace pak probíhá pouze pomocí protokolu μTP.



Obrázek 4.2: Paket s požadavkem get_peers, zasílaný na synchronizační server.

Vzhledem k tomu, že uzel zasílá požadavek k již známému klíči, bude odpověď od serveru vždy obsahovat informaci alespoň o jednom aktivním uzlu. Tím je právě uzel, který tento požadavek na server zaslal [9]. Na obr. 4.3 lze vidět, že v odpovědi od synchronizačního serveru se nachází pouze jediný uzel.

Synchronizační server se také chová jako STUN[2] server a umožňuje tak uzlům vytvořit přímá spojení, i pokud se nachází za systémem NAT[3](Obr. 4.4, cesta B)



Obrázek 4.3: Paket s odpovědí synchronizačního serveru na požadavek get_peers.

4.1.3 Tabulka DHT

Klient využívá distribuovanou hashovací tabulku k šíření informací o sobě a k získání informací o ostatních uzlech vlastnící stejný klíč. Klient zaregistruje informace o sobě samém rozesláním zprávy ostatním uzlům, ve tvaru SHA1(Secret):IP:port. Pro získání seznamu uzlů k danému klíči zašle požadavek na DHT ve tvaru SHA1(Secret). Samotná DHT tabulka je uložena na serverech router.bittorrent.com a router.utorrent.com.

4.1.4 Známé uzly

Poslední metodou je možnost zadání předdefinovaných uzlů. Jedná se o metodu, která je nejhůře rozpoznatelná. Uživatel může k předvolbě sdílení přidat seznam s kombinacemi IP adres a čísel portů. Uzly nacházející se na tomto seznamu budou kontaktovány přímo, bez jakékoliv nutnosti zasílání paketu s BSYNC hlavičkou a zprávou ping.

4.2 Přenos dat

4.2.1 Přímá komunikace

Odesílající uzel se pokusí přímo komunikovat s uzlem, kterému mají být data doručena (na obr. 4.4 označeno cestou A). Pokud se jedná o komunikaci uskutečňovanou mimo lokální síť, je tato komunikace standardně šifrována. Při komunikaci v lokální síti lze toto šifrování vypnout v nastavení aplikace.

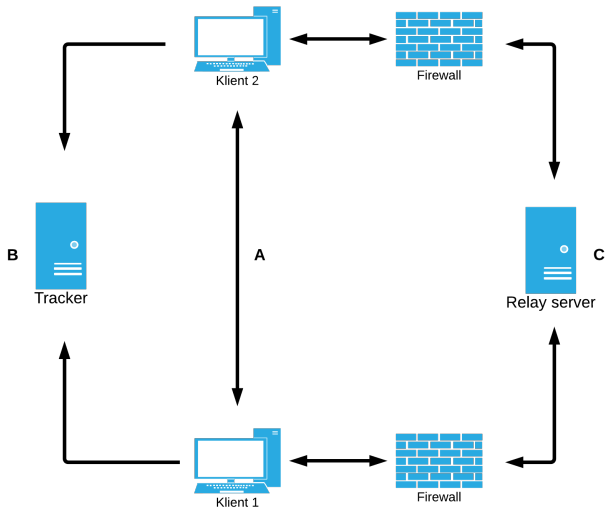
[2]STUN – síťový protokol dovolující koncovým uživatelům zjistit jejich IP adresu a port, který jim přidělil systém NAT [8].

[3]NAT – systém pro překlad síťových adres.

4.2.2 Přepojovací server

V případě že přímá komunikace mezi uživateli není možná, například z důvodů blokování komunikace firewallem, lze toto omezení obejít použitím přepojovacího serveru (obr. 4.4, cesta C). Přepojovací server se nachází na adrese r.usyncapp.com, která se překládá na jednu z těchto IP adres[4]:

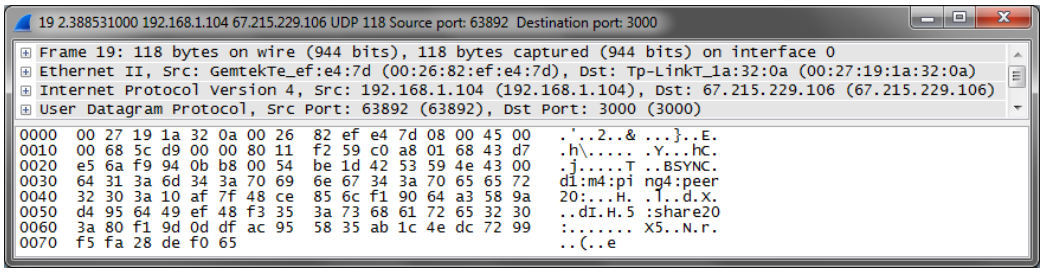
- 67.215.229.106
- 67.215.231.242
- 66.63.177.26



Obrázek 4.4: Možnosti komunikace uzlů.

Pakety jsou odesílány na přepojovací server pomocí transportního protokolu UDP na port 3000 a obsahují ping zprávy. V těchto ping zprávách je obsaženo PeerID a ShareID, jež je odvozeno od klíče.

Po úvodní komunikaci přepojovací server vyjedná datový komunikační spoj se vzdáleným uživatelem. Poté, co se dohodne komunikace se vzdáleným uživatelem, odešle se paket s 160 bitovým veřejným klíčem a může začít šifrovaný přenos dat.



Obrázek 4.5: Zaslání požadavku na přepojovací server.

[4]Informace o IP adresách lze nalézt na http://config.usyncapp.com/sync.conf

4.3 Shrnutí

V této kapitole byly vysvětleny metody, které BTSync používá k objevování nových uzlů. Dále zde byly popsány možnosti, jakými mohou dva uzly spolu komunikovat. Tyto znalosti v dalších kapitolách využijeme například při navrhování systému na monitorování provozu aplikace BTSync.

Kapitola 5

Implementace nástroje

V předchozích kapitolách byly uvedeny informace o zprávách a typech komunikacích, které aplikace BTSync používá během svého provozu. Pomocí těchto zjištěných poznatků pak byl vytvářen nástroj pro monitorování provozu aplikace BTSync.

Finální verze tohoto nástroje se skládá ze dvou částí. První část nástroje sleduje provoz na síti a podle pravidel zaznamenává podezřelé pakety. Druhý nástroj pak slouží k prezentaci všech zaznamenaných paketů.

V dalších částech této kapitoly budou oba nástroje detailněji rozebrány a popsána jejich implementace.

5.1 Nástroj pro zachytávání paketů

Nástroj pro detekci komunikace BTSync je napsán v jazyce Python. Ke zpracování paketů využívá knihovnu scapy[1].

Při spuštění nástroje je možné určit, zda bude detekce probíhat on-line nebo bude zdrojem dat ke zpracování soubor .pcap s nasnímaným síťovým provozem. Pro monitorování komunikace on-line je potřeba nástroj spouštět se správcovskými právy.

Pro zadání vstupu je nutné zadat jeden z těchto parametrů:

- -i, --interface=rozhraní – Vstupem je zadané rozhraní.
- -f, --file=soubor – Bude zpracováván zadaný pcap soubor.

Dále je také nutné zadat, kam se data s detekovanou komunikací budou ukládat nebo vypisovat. Na výběr je jedna ze tří možností. První z nich je ukládání dat do CSV souboru, druhou možností je vypisování dat na terminálu, obr. 5.1. Třetí možností je ukládat záznamy o detekovaném paketu do databáze.

Výstup programu lze zadat pomocí těchto parametrů:

- -o, --outfile=soubor – Data budou zapisována do souboru CSV.
- -d, --database – Data budou zapisována do databáze.
- -t, --terminal – Výstup dat bude prováděn na terminál.

[1] Více informací o této knihovně lze najít na http://www.secdev.org/projects/scapy/

```
martin@lenovoE550 ~/Dokumenty/BTSYNC/latest $ python bsync.py -f pcap/btsync.pcap -t
2014-10-13 21:30:44, 1cc1de8cfb5a, 147.229.186.178, 17061, b8af677ed060, 54.225.92.50, 3000, UDP, 66, b_data
2014-10-13 21:30:44, 1cc1de8cfb5a, 147.229.186.178, 17061, b8af677ed060, 157.7.205.115, 32822, UDP, 62, b_ack
2014-10-13 21:30:44, b8af677ed060, 54.225.92.50, 3000, 1cc1de8cfb5a, 147.229.186.178, 17061, UDP, 62, b_ack
2014-10-13 21:30:45, b8af677ed060, 67.215.231.242, 3000, 1cc1de8cfb5a, 147.229.186.178, 39825, TCP, 108, b_relay
2014-10-13 21:30:45, 1cc1de8cfb5a, 147.229.186.178, 39825, b8af677ed060, 67.215.231.242, 3000, TCP, 66, b_relay
```

Obrázek 5.1: Ukázka výpisu zaznamenaných paketů na terminál.

Posledním možným parametrem, který lze zadat je parametr, který stáhne aktuální konfiguraci synchronizačních a přepojovacích serverů. V konfiguraci se nacházejí IP adresy a porty, na které se klienti pokoušejí připojit.

- -u, --update

5.1.1 Rozpoznávání komunikace

První krokem, kterým každý paket prochází je odfiltrování irelevantních paketů. To jsou takové pakety, které nemají IPv4 adresu a jejich transportním protokolem není TCP nebo UDP. Pokud pakety tímto filtrem projdou, jsou předány rozhodovacímu algoritmu, obr. 5.2.

Ten porovnává IP adresy, porty a také obsah některých paketů se známými adresami serverů, adresami podezřelých uzlů a také s řetězci, které se v paketech BTSync komunikace nacházejí.

Pokud dojde k rozpoznání takové komunikace, vytvoří se třída podezřelého, která uchovává informace o IP adrese a portech, na kterých podezřelý uzel komunikoval. Tyto informace se pak využívají pro detekci BTSync komunikace mezi dvěma uzly. Každá třída obsahuje také časovač, který je každým výskytem komunikace daného uzlu obnovován. Pokud dojde k vypršení časovače, je neaktivní uzel odstraněn.

Pokud algoritmus rozpozná, že zdrojová nebo cílová IP adresa a port patří jednomu ze synchronizačních serverů, dochází k dalšímu zpracování těchto paketů. Nejprve se zkoumá velikost těchto paketů. Menší velikost paketů než 45B značí, že se jedná o obecné informační pakety. K zjištění typu informačního paketu se porovnává první Byte paketu s typy μTP zpráv.

Pokud je velikost paketů vetší než určených 45B, hledají se v obsahu paketů řetězce *get_peers* a *peers*[2]. Při nalezení řetězce *get_peers* se uloží informace o rozpoznaném paketu a pokračuje se na další paket. Pokud dojde k rozpoznání řetězce *peers*, jsou adresy obsažené v seznamu dostupných uzlů uloženy do seznamu podezřelých uzlů.

U paketů zasílaných nebo příchozích z multicastové adresy *239.192.0.0* a portu 3838 se v obsahu paketu vyhledává řetězec *BSYNC*. Pokud ho paket obsahuje, uloží se informace o rozpoznaném paketu a pokračuje se na další paket.

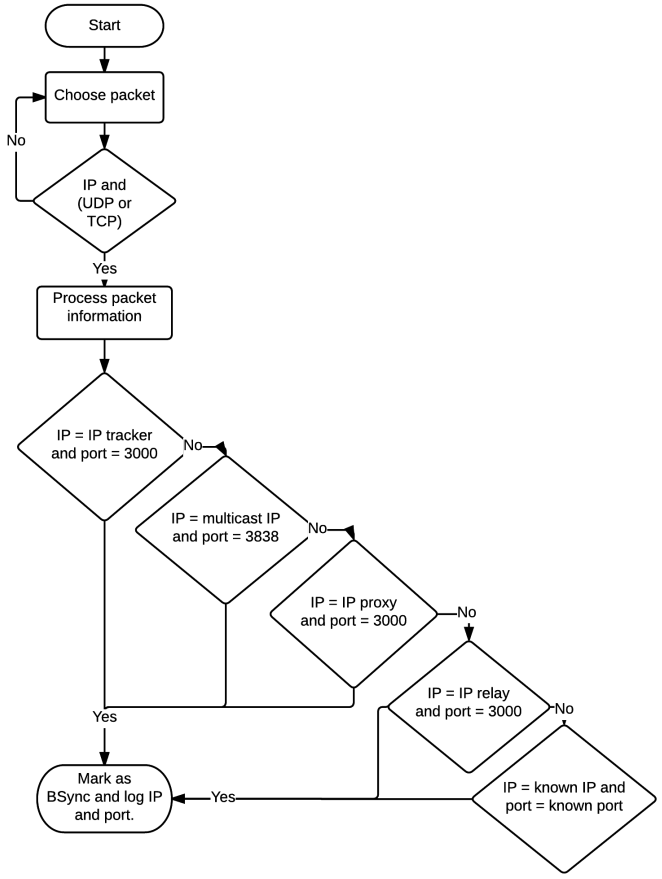
Dále se pak kontroluje zda adresa odesílatele nebo příjemce paketu neodpovídá některé z adres přepojovacího serveru nebo proxy serveru. U těchto komunikací se jiné údaje nekontrolují a ukládají se pouze informace o rozpoznaném paketu.

Pokud ani jedna z výše uvedených podmínek neplatí, kontroluje se, zda se v paketech nenachází řetězec *BSYNC*. Jestliže se v něm nachází vyhledává se v paketu řetězec *ping*. Pokud se v něm nachází i tento řetězec je paket

Posledním porovnávacím pravidlem je zjištění, zda se adresa odesílatele nebo adresa příjemce nenachází v seznamu s podezřelými uzly. Jestliže se zjistí, že jedna z těchto adres se nachází v seznamu s podezřelými uzly, kontroluje se obsah takových paketů a vyhledávají

[2]Jelikož by mohlo docházet k nesprávnému vyhodnocení, vyhledává se v paketech řetězec *m5:peers*.

se v něm řetězce vyskytující se v paketech DHT komunikace. Těmi jsou *find_node*, *get_peers*, *ping* a *nodes*. Pokud se v něm tyto řetězce nenacházejí zkontroluje se typ v μTP hlavičce.



Obrázek 5.2: Zjednodušený rozhodovací algoritmus nástroje pro monitorování.

Třída podezřelých

Pro ukládání informací o podezřelých IP adresách a portech, na kterých probíhá BTSync komunikace je v nástroji vytvořena třída *SuspectClass*.

Každá třída má čtyři instanční proměnné. Tyto proměnné v sobě mají uloženou IP adresu, kterou podezřelý uzel používá, množinu TCP portů a množinu UDP portů, kterých uzel při komunikaci využívá a proměnnou časovače.

Součástí třídy SuspectClass je pět metod. Těchto pět metod se stará o přístup k instančním proměnným objektu a aktualizaci hodnoty časovače.

Třída časovač

Jelikož uzly zůstávají aktivní pouze jenom na několik hodin a poté se odpojí, je potřeba vytvořit mechanismus, který po určité době neaktivní uzly ze seznamu podezřelých odstraní. Z tohoto důvodu se v nástroji nachází třída *TimerClass*.

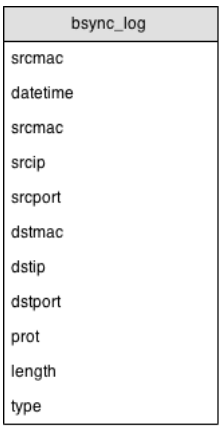
Úkolem třídy TimerClass je čítat čas pro každý uzel a pokud čítač dosáhne prahové hodnoty, pak je tento uzel ze seznamu s podezřelými uzly odstraněn.

Časovač je naprogramován jako vlákno, které běží v nekonečném cyklu a je na sekundu uspáváno. Po probuzení se čítač o jedničku sníží. Pokud čítač není obnoven na výchozí hodnotu a dosáhne nuly, je uzel pro který časovač běží odstraněn. Obnovování čítače je zajištěno metodou *updatecount*, ta je volána v případě, že se v síťovém provozu vyskytne paket s IP adresou a portem podezřelého uzlu.

Databáze

Jednou z možností kam ukládat data s detekovanou komunikací je databáze. K ukládání dat do databáze je potřeba vytvořit tabulku bsync_log. Tabulku, obr. 5.3, tvoří deset sloupců. Primárním klíčem je sloupec *pid*, který slouží jako identifikace paketu v tabulce.

K uložení IP adres do databáze je využito funkce *INET_ATON*, která IP adresy převádí na celá čísla. Při vypisování dat z databáze je pak použito opačné funkce *INET_NTOA*. Ostatní hodnoty jsou do databáze ukládány nezměněny.

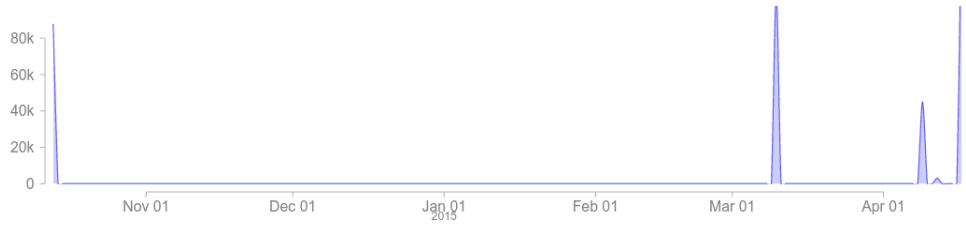


Obrázek 5.3: Tabulka pro ukládání dat do databáze.

5.2 Vizualizace zachycených dat

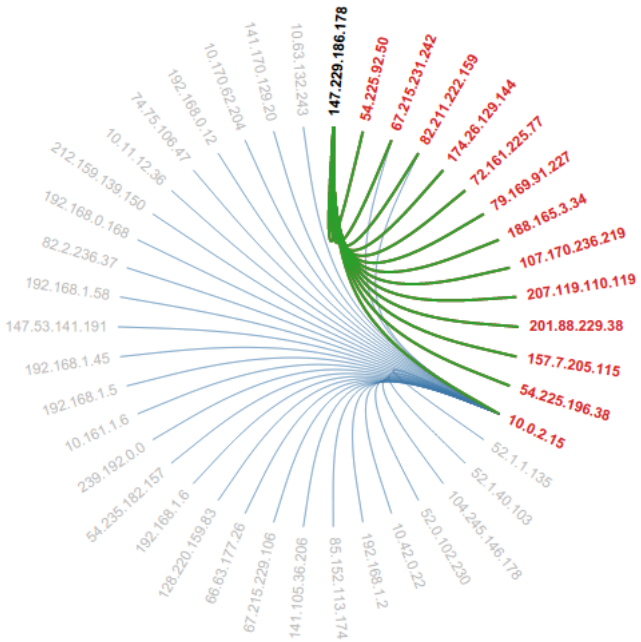
Samotný monitorovací nástroj produkuje data pouze v textové podobě. Textová data avšak sama o sobě neposkytují příliš informací, nebo jich poskytují až příliš a je složité v nich najít to důležité. A právě z rychlého poskytnutí důležitých informací byly k monitorovacímu nástroji vytvořeny webové stránky. Základ těchto stránek je napsán v jazyce PHP. K vizualizaci dat do grafů, tabulek a map je použit jazyk JavaScript.

Webové stránky jsou rozděleny do tří částí, které prezentují různé informace. První částí je úvodní strana. V horní polovině se nachází graf zobrazující velikost přenášených paketů v čase, obr. 5.4. Z tohoto grafu lze vyčíst, kdy docházelo k přenosům dat a kolik dat bylo přeneseno. K vytvoření tohoto grafu byla použita javacsriptová knihovna MetricsGraphics.



Obrázek 5.4: Graf s velikostí přenesených dat v čase.

Pod grafem toků dat se nachází mapa komunikací mezi jednotlivými uzly. Najetím myši na IP adresu uzlu se zvýrazní IP adresy, se kterými vybraný uzel komunikoval, obr. 5.5. Černě je označen vybraný uzel, červeně pak uzly, které s adresou komunikovaly. Tato mapa nám poskytuje rychlý přehled o tom, kdo s kým komunikoval. K vytvoření mapy spojení byla použita javasciptová knihovna D3.

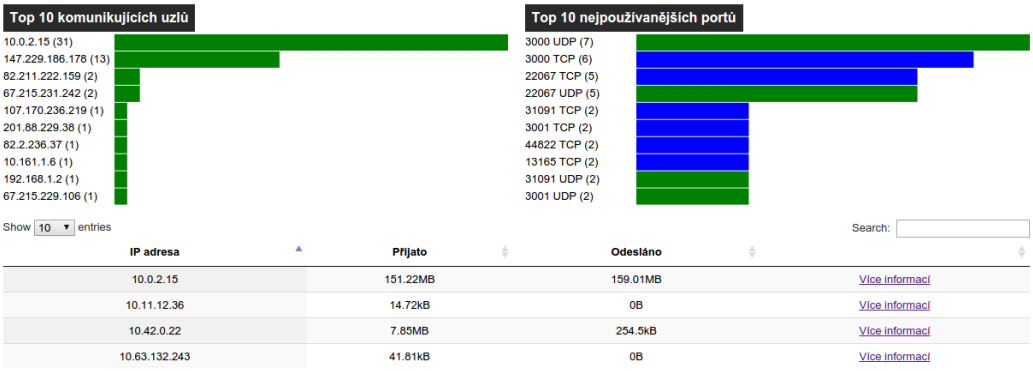


Obrázek 5.5: Mapa toků dat mezi uzly.

Na konci úvodní webové stránky se nachází dva grafy a tabulka, obr. 5.6. V prvním grafu je zobrazeno deset nejaktivnějších uzlů (s největším počtem navázaných spojení), v závorce za IP adresou je uveden počet navázaných spojení (s kolika IP adresami probíhala komunikace).

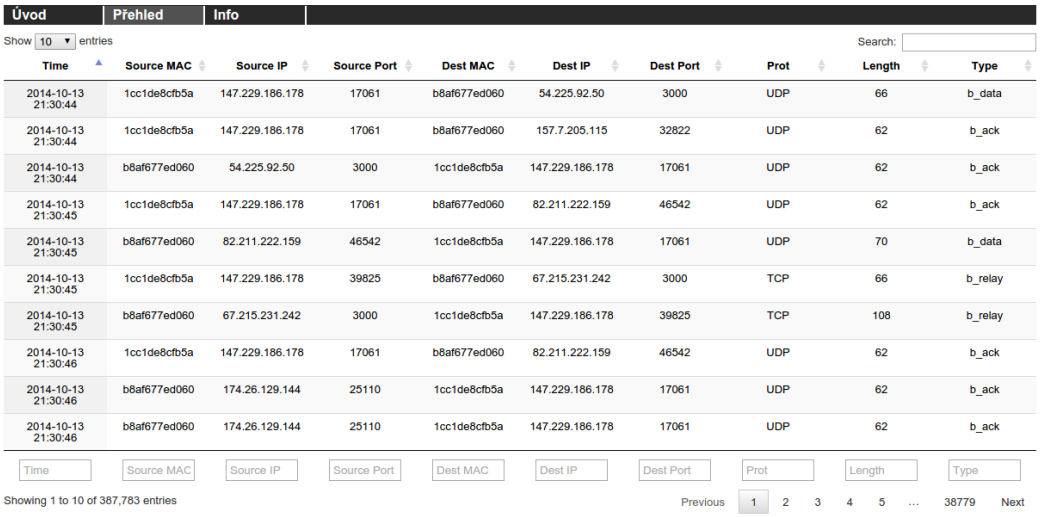
Ve druhém grafu je vypsáno deset nejvyužívanějších portů. V závorce za číslem portu je uveden počet paketů, ve kterých se daný port nacházel.

Pod těmito grafy se pak ještě nachází tabulka, ve které jsou uvedeny všechny IP adresy, nalezené v zachycené komunikaci u každé IP adresy je pak vypsána velikost dat přijatá danou IP adresou a kolik dat bylo odesláno z dané IP adresy. V posledním sloupci se pak nachází odkaz, který nás přesměruje do třetí části webových stránek.



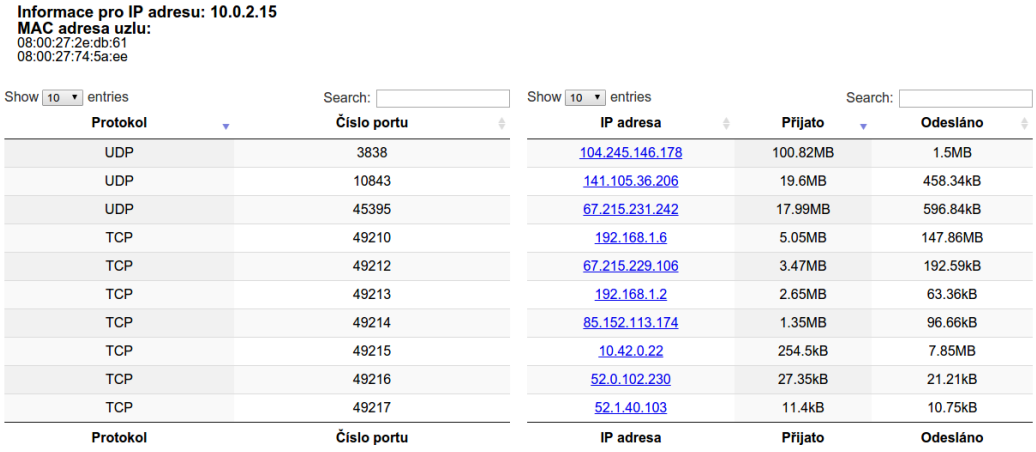
Obrázek 5.6: Grafy deseti nejaktivnějších uzlů a deseti nejpoužívanějších portů.

Druhou část webových stránek tvoří tabulka se všemi zachycenými pakety. V tabulce lze vyhledávat a tím také filtrovat zobrazované informace. Jednotlivé sloupce tabulky lze také seřazovat. Tato tabulka nám poskytuje možnost vyhledávat doplňující informace, které nás zajímají. K vytvoření tabulky bylo použito jQuery rozšíření Datatables.



Obrázek 5.7: Celkový přehled všech zachycených dat s možností vyhledávání.

Poslední třetí část webu tvoří stránka, která slouží k vypisování detailnějších informací k jednotlivým IP adresám. Na stránce, obr. 5.8, se vypisuje zvolená IP adresa, MAC adresa, případně MAC adresy, které zvolenou IP adresu používaly. Dále se tu pak nachází dvě tabulky. V jedné jsou uvedeny všechny porty využívané zvolenou IP adresou a v druhé tabulce je pak seznam IP adres, se kterými probíhala komunikace a počet přenesených dat.



Informace pro IP adresu: 10.0.2.15
MAC adresa uzlu:
08:00:27:2e:db:61
08:00:27:74:5a:ee

Show 10 entries Search:

Protokol	Číslo portu
UDP	3838
UDP	10843
UDP	45395
TCP	49210
TCP	49212
TCP	49213
TCP	49214
TCP	49215
TCP	49216
TCP	49217

Show 10 entries Search:

IP adresa	Přijato	Odesláno
104.245.146.178	100.82MB	1.5MB
141.105.36.206	19.6MB	458.34kB
67.215.231.242	17.99MB	596.84kB
192.168.1.6	5.05MB	147.86MB
67.215.229.106	3.47MB	192.59kB
192.168.1.2	2.65MB	63.36kB
85.152.113.174	1.35MB	96.66kB
10.42.0.22	254.5kB	7.85MB
52.0.102.230	27.35kB	21.21kB
52.1.40.103	11.4kB	10.75kB

Obrázek 5.8: Výpis informací o uzlu.

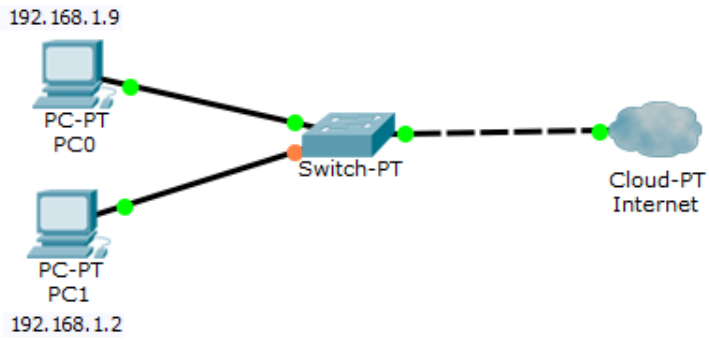
Kapitola 6

Testování

K ověření funkčnosti nástroje pro detekci provozu aplikace BTSync bylo provedeno jeho testování. Záznamy pro testování byly nasbírány z provozu v lokální síti. K nasbírání dat do pcap souborů byl použit program Wireshark. Tyto záznamy pak byly použity k provádění testů s monitorovacím nástrojem. V následujícím textu pak bude podrobně představeno pracoviště, ve kterém byla data zachytávána a testy, které byly prováděny.

6.1 Popis pracoviště

Pro sběr dat bylo použito zapojení na obr. 6.1. V něm běžela na zařízení PC2 běžela aplikace BTSync, na zařízení PC1 byl spuštěn virtuální počítač, na kterém běžela kromě aplikace BTSync také ještě aplikace qBittorrent a Skype. Ty se snažily generovat síťový provoz, který by mohl vést k falešným detekcím. K PC1 byl ještě navíc připojen mobilní telefon přes vytvořené Wi-Fi připojení. Na mobilním telefonu s operačním systémem Windows Phone bežela mobilní aplikace BTSync. Toto zapojení má ovšem jednu nevýhodu – nedá se v něm generovat takový síťový provoz, jaký lze očekávat ve větších sítích nebo na páteřních linkách, a tím otestovat jak by nástroj fungoval pod zátěží. Sběr dat probíhal na systému Linux Mint, na PC1 za pomoci programu Wireshark. Pakety byly snímány po dobu 5 minut. Během této doby běželo na virtuálním počítači stahování torrentu linuxové distribuce Ubuntu a BTSync synchronizovala data od neznámých uzlů. Mobilní aplikace běžící na mobilním telefonu se pokoušela sesynchronizovat s PC2.



Obrázek 6.1: Použité zapojení pro sběr dat použitých k testování.

6.2 Popis testování

Nástroj byl testován za pomoci nasbíraných dat v pcap souboru. V tabulce 6.1 je vidět, že celkem bylo nasnímáno 214181 paketů a z toho 97319 z nich bylo označeno za BTSync pakety.

Paketů	**Detekovaných paketů**
214 181	97 319

Tabulka 6.1: Tabulka s celkovým počtem paketů a počtem detekovaných paketů.

Zkoumáním detekovaných paketů se došlo k závěru, které jsou zobrazeny v tab. 6.2. Tyto výsledky jsou pozitivní. Přesnost nástroje je ovšem vykoupena nutností zkoumat obsah všech paketů. To způsobuje, že nástroj není příliš rychlý.

Správných detekcí	**Nesprávných detekcí**	**Falešných detekcí**
97 319	0	0

Tabulka 6.2: Tabulka s počtem správně detekovaných paketů, nesprávně detekovaných a falešně detekovaných.

Během testování byly sledovány také údaje o vytížení procesu a paměťové nároky a rychlost zpracování dat nástrojem.

Tyto údaje jsou shrnuty v tabulce 6.3. Mód nástroje offline znamená, že vstupem byl pcap soubor. V tomto případě jím byl soubor o velikosti 169,5 MB. Mód online 1 označuje stav nástroje během malého síťového provozu. Mód online 2 značí použití nástroje v plném provozu.

Mód nástroje	**% vytížení CPU**	**Paměťové nároky**	**Rychlost zpracování**
Offline	100 %	32 MiB	158 s
Online 1	6 %	22 MiB	–
Online 2	100 %	30 MiB	–

Tabulka 6.3: Nároky nástroje pro monitorování na hardware.

Kapitola 7

Závěr

Cílem bakalářské práce bylo provedení analýzy síťového provozu aplikace BTSync a vytvoření systému, který bude sloužit k monitorování tohoto provozu a bude poskytovat vhodnou prezentaci získaných výsledků.

Jelikož aplikace BTSync využívá ke svému fungování sítě P2P, byla v úvodu práce popsána architektura P2P sítí. Ve druhé kapitole byl také popsán protokol BitTorrent a představena samotná aplikace BTSync a její protokol. Ve třetí kapitole byly rozebrány některé z možných přístupů k detekci P2P komunikace. Ve čtvrté kapitole byla provedena analýza síťového provozu aplikace BTSync a jsou zde uvedeny typy komunikací. Pátá kapitola obsahovala popis návrhu a implementace nástroje pro monitorování síťového provozu aplikace BTSync. V závěru práce jsou uvedeny experimenty, které byly s nástrojem prováděny a výsledky těchto experimentů.

Během provádění testování bylo zjištěno, že vytvořený nástroj je poměrně přesný a neprodukuje žádné falešné detekce. Jeho nevýhodou ovšem je, že během zpracovávání většího množství paketů nadměrně zatěžuje procesor počítače.

Možností, jak nástroj vylepšit by bylo jeho přeprogramování do jazyka C/C++. Díky čemuž by došlo ke zrychlení vyhodnocování jednotlivých paketů a bylo by tak možné nástroj používat i ve větších sítích.

Literatura

[1] Buford, J.; Yu, H.; Lua, E.: *P2P networking and applications*. Morgan Kaufmann Publishers, 2008, iSBN 978-0-12-374214-8.

[2] Cohen, B.: The BitTorrent Protocol Specification. [Online], 2008, [cit. 2015-04-02].
URL http://www.bittorrent.org/beps/bep_0003.html

[3] Farina, J.; Scanlon, M.; Kechadi, M.-T.: BitTorrent Sync: First Impressions and Digital Forensic Implications. *Digital Investigation*, ročník 11, Supplement 1, č. 0, 2014: s. S77 – S86, ISSN 1742-2876, proceedings of the First Annual {DFRWS} Europe.
URL http://www.sciencedirect.com/science/article/pii/S1742287614000152

[4] Gong, Y.: Identifying P2P users using traffic analysis. [Online], 2005, [cit. 2015-04-02].
URL http://www.symantec.com/connect/articles/identifying-p2p-users-using-traffic-analysis

[5] Jalsovszky, Z.: *Rozpoznání uživatelů P2P sítí na základě analýzy síťového provozu*. Bakalářská práce, FIT VUT v Brně, 2009.

[6] Letý, P.: *Detekce peer-to-peer komunikace*. Bakalářská práce, FIT VUT v Brně, 2014.

[7] Norberg, A.: μTorrent Transfer Protocol. [Online], 2009, [cit. 2015-04-02].
URL http://www.bittorrent.org/beps/bep_0029.html

[8] Rosenberg, J.; Mahy, R.; Matthews, P.; aj.: Session Traversal Utilities for NAT (STUN). [Online], Říjen 2008, [cit. 2015-04-02].
URL https://tools.ietf.org/html/rfc5389

[9] Scanlon, M.; Farina, J.; Kechadi, M.-T.: BitTorrent Sync: Network Investigation Methodology. In *Proceedings of 9th International Conference on Availability, Reliability and Security (ARES 2014)*, Fribourg, Švýcarsko: IEEE, Září 2014, s. 21–29.

[10] Starigazda, M.: *Detekce sítí P2P pomocí Netflow*. Bakalářská práce, FIT VUT v Brně, 2013.

[11] WWW stránky: BitTorrent Sync Technical Specs. [Online], [cit. 2015-01-29].
URL www.getsync.com/tech-specs

[12] WWW stránky: BitTorrent Sync Help Center. Key structure and flow. [Online], [cit. 2015-04-02].
URL http://sync-help.bittorrent.com/customer/portal/articles/1628254-key-structure-and-flow

[13] WWW stránky: BitTorrent Sync Help Center. What is Sync? [Online], [cit. 2015-04-02].
URL http://sync-help.bittorrent.com/customer/portal/articles/1574733-what-is-sync-

Příloha A

Obsah CD

Přiložené CD obsahuje následující soubory:

- Bakalářskou práci ve formátu PDF
- Zdrojové soubory této práce
- Zdrojové soubory monitorovacího nástroje
- Zdrojové soubory vizualizačního nástroje
- Testovací data s BitTorrent Sync komunikací